

A SIMULATOR FOR CONTINUOUS AGENT-BASED MODELLING

J. DUGGAN*, National University of Ireland, Galway, Ireland.

ABSTRACT

This paper describes a simulation environment that can be used to integrate population-level dynamics with those occurring at an individual, or agent-based, level. The benefit of this approach is that individual agent behaviour may be mapped at a detailed level, using differential equations, and aggregated over the entire population in order to determine population-level dynamics. Furthermore, individual agents can interact with one another, in terms of a social network structure. The environment is firmly grounded in the system dynamics approach, and, unlike conventional agent-based simulation environments, programming is not required in order to specify agent interactions and behaviours. The approach is validated by using the classic SIR model of contagion.

Keywords: System Dynamics, Agent Based Modelling, Simulation, SIR Models

INTRODUCTION

This paper proposes an extension to the System Dynamics (SD) method in order to provide a novel way for modeling multi-agent systems. Gilbert and Troitzsch (2005) comment that “a natural way of programming agents is to use an object-oriented programming language,” and frameworks such as RePast (North et al. 2006), AnyLogic (Borschev and Filippov 2004), and NetLogo are commonly used to achieve this goal. A characteristic of the ABM approach is the focus on agent heterogeneity, namely, identifying the differences in agents and simulating their interactions and behaviour over time. System Dynamics is an alternative approach to agent-based simulation, and employs as a robust and well-defined methodology to model the behaviour of decision making entities. The resulting simulations are run in continuous time, and, do not require programming expertise on behalf of the modeler. However, a disadvantage of the system dynamics approach regards the scale of agent models. Current system dynamics tools are not amenable to the construction of large scale agent societies, and so the possibilities for extending the heterogeneity of models is limited. The approach presented here addresses provides an approach and technology that allows large scale agent models to be built, based on sets of differential equations.

SYSTEM DYNAMICS

The systems approach to problem solving has many strands and influences, and originally emerged as a reaction to the reductionism advocated by the traditional “divide and conquer” approach to science, namely, that in order to understand a complex system, one must take it apart and understand its constituent parts. Systems thinking involves taking a holistic approach to problem solving, by identifying the manner in which systems interact in order to uncover insights regarding overall systems behaviour. System Dynamics (SD) is one of the most widely used systems approaches in the world, and was created and pioneered by Jay W. Forrester

* Corresponding author address: Jim Duggan, Department of Information Technology, NUI, Galway, Ireland.

(Forrester 1958, Forrester 1961) as a means of simulating the behaviour of social systems, explaining that behaviour, and crafting effective long term policies (Lane 2006). Sterman (2000) defines SD as “a perspective and a set of conceptual tools that enable us to understand the structure and dynamics of complex systems” and, because of its underlying mathematical foundation, SD is also “a rigorous modeling method that enables us to build formal computer simulations of complex systems and use them to design more effective policies and organizations.”

Forrester’s key insight followed on his study of manufacturing supply chains, and his observation that the chaos apparent in inventory levels throughout the supply chain could not be explained solely by external effects. He demonstrated that many problems were in caused internally, by the decisions of managers who were not fully aware of the feedback structures that were present in the overall system. Forrester argued that feedback is a critical component of human decision making, as many management decisions lead “to a course of action that changes the state of the surrounding system and gives rise to new information on which future decisions are made” (Forrester 1969).

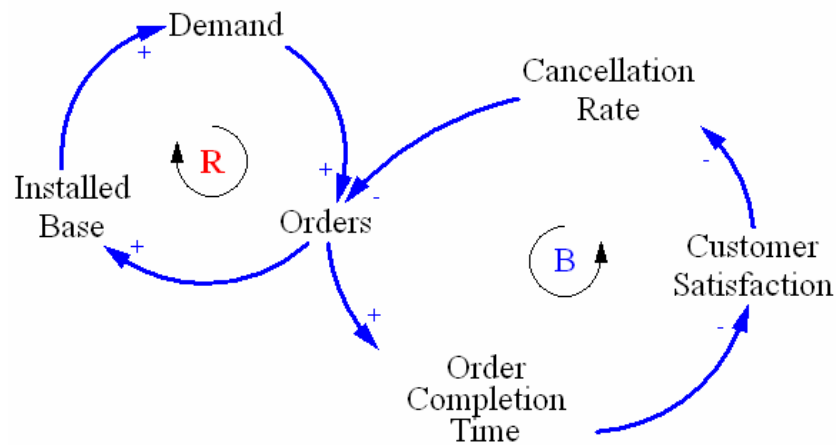


FIGURE 1 Limits to growth in a feedback system

Decision makers who do not take account of the feedback view often encounter policy resistance, where “well-intentioned policies are delayed, diluted and defeated by the unforeseen reactions of other people” (Sterman 2000). A simple example of this is captured in Figure 1, which shows two feedbacks. The first, a *reinforcing feedback*, models the “word of mouth” phenomenon, where demand fuels orders, which in turn increase the installed base, and this leads to further demand. The second loop, a *balancing feedback*, acts to dampen this growth, because capacity limitations in the system mean that as orders rise, so to does the order completion time, which reduces customer satisfaction, and this in turn leads to order cancellations. It is the interaction of these loops that determines how the system behaves over time, and while sales may grow exponentially at first, any serious capacity constraint will cause this growth to flatten over time. A possible solution to this would be to increase capacity early (a proactive strategy) in the product’s life cycle, rather than increasing the capacity at a later stage (a reactive strategy).

Figure 1 illustrates how feedback structures can be captured qualitatively, using causal loop diagrams. In order to formally model dynamic systems, these feedbacks must be represented quantitatively, through *stock and flow* diagrams. Flows are formal expressions of policy, where equations are formulated to capture the key decision rules that control the rates of flow through a system. Stocks are accumulations (e.g. number of employees, balance in a bank account, number of people queuing), and the completed models are run as a set of non-linear differential equations. However, a disadvantage of current system dynamics tools is that they do not provide the mechanism to construct large-scale agent models. The approach presented here addresses these shortcomings by providing an approach and technology that allows large scale agent models to be built, based entirely on sets of differential equations.

SYSTEM DESIGN

Figure 2 illustrates the conceptual design, and this is an adaptation of Sterman's (2000, p.515) representation of how decision rules govern the rates of flow in systems. The central idea is that each agent is represented by an individual stock and flow structure. An agent changes its state by processing information cues: in this generic model, these cues can be based on the agent's own state, information from other agents, and information from the aggregate system state. This aggregate system state is a summation of the individual rates of change for each agent in the population, and so feedback exists between the aggregate level and the individual agent level.

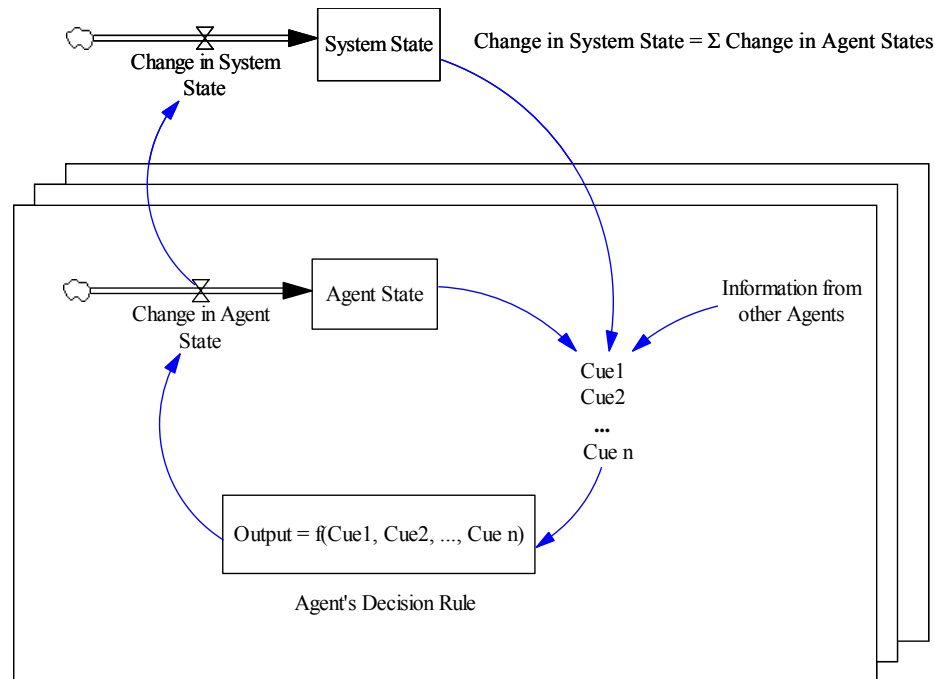


FIGURE 2 Conceptual design for the continuous agent based modeller

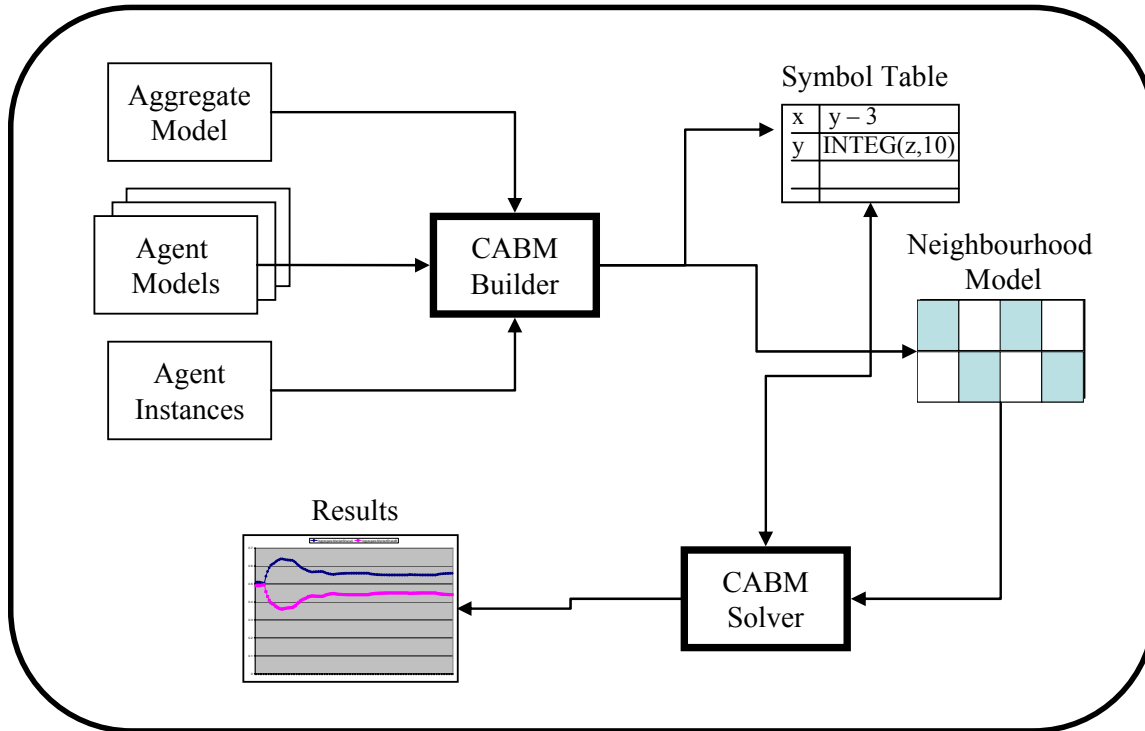


FIGURE 3 Architecture for the continuous agent based modeller

The system architecture is captured in Figure 3. The major software components are:

CABM Builder. This takes as input three model types, and creates: (1) a symbol table containing each equation to be solved, including stocks, flows and auxiliaries, and (2) a neighbourhood model that places each agent in a grid-like structure so that information cues from neighbours can be taken into account for an agent's decision making. The number of equations created is a function of the number of agents. For example, if each agent is represented by thirty equations, and there are one hundred agents, then there will be three thousand equations in the symbol table (excluding those equations that are specified as part of the aggregate model).

CABM Solver. This solves all numerical equations contained in the symbol table, and also has special-purpose routines that can aggregate variables in the model, and calculate neighbourhood values for each individual agent.

There are three main types of input for the CABM Builder:

The aggregate model, which corresponds to the "System State" element of Figure 2. These are the stocks and flows that capture the aggregate dynamics for the system of interest. The key states in this model will change based on an aggregation of all the changes at the agent (or lower) level of the model.

The agent models, which express important agent heterogeneities in the system under consideration. Each agent model can have different formulations for key decision equations.

The agent instances, which specify how many of each agent are to be created for a simulation run, and also can be used to vary the specified parameter values of individual agents.

In order for the system to work, three categories of mapping must be achieved when the models are combined. The first of these is mapping from the detailed level to the aggregate level, where a high-level flow is formulated based on an aggregation of lower level, or agent, flows. A stock and flow is identified as an aggregate by including the tag “<is_aggregate>” as part of its definition (see Figure 4). For an aggregate flow, no equation is specified: instead, a purpose-built function – called AGGREGATOR() – is invoked, and during the simulation run this function will aggregate all the relevant agent flows.

```
<stock>
  <name>Aggregate.Susceptible</name>
  <is_aggregate>true</is_aggregate>
  <init>0.0</init>
  <outflow>Aggregate.InfectionRate</outflow>
</stock>

<flow>
  <name>Aggregate.InfectionRate</name>
  <is_aggregate>true</is_aggregate>
  <equation>AGGREGATOR()</equation>
</flow>
```

FIGURE 4 Defining stocks and flows at an aggregate level

At the agent level, any flow that must aggregate to a higher level is specified with the tag “<is_subflow>”, and the super flow, which is the flow that it aggregates to, is specified (Figure 5). In this example, the string “\$NAME\$” will be replaced by the specified agent name when the model is created. Stocks at the agent level are defined in a similar manner (Figure 6). Furthermore, if a rate at the agent level is procedurally complex, a callout routine can be written (in C#) to evaluate it. Rates at the agent level that mirror an aggregate rate are usually programmed to flow over one time step, and so state switching occurs at discrete points in the simulation.

```
<flow>
  <name>$NAME$.RecoveryRate</name>
  <is_subflow>true</is_subflow>
  <super_flow>Aggregate.RecoveryRate</super_flow>
  <equation>DELAYFIXED($NAME$.InfectionRate, $NAME$.RecoveryDelay,0)</equation>
</flow>
```

FIGURE 5 Defining flows at an agent level

```

<stock>
  <name>${NAME$.Recovered}</name>
  <is_substock>true</is_substock>
  <super_stock>Aggregate.Recovered</super_stock>
  <init>${RECOVERED_INIT}</init>
  <inflow>${NAME$.RecoveryRate}</inflow>
  <capture_state>true</capture_state>
  <value_if_true>300</value_if_true>
</stock>

```

FIGURE 6 Defining stocks at an agent level

The final mapping relates to agent-to-agent communications, where an agent uses information from other agents in order to arrive at a decision. In order to facilitate this, the agents are modelled as a society in a grid-like structure. In figure 7, a society of 100 agents are shown, where each of these has a similar stock and flow structure. The purpose-built function NEIGHBOURHOOD_AVERAGE() will find the average for a given model variable from across all of its immediate neighbours, and this value that then be used as an important cue in an agent's decision making process. This value would play an important part in triggering whether or not an individual agent may change its state (for example, from susceptible to infected in a model of contagious disease).

```

<auxiliary>
  <name>${NAME$.NeighboursAvg}</name>
  <equation>NEIGHBOURHOOD_AVERAGE(${NAME$, ${NAME$.Infected})</equation>
</auxiliary>
<flow>
  <name>${NAME$.InfectionRate}</name>
  <is_subflow>true</is_subflow>
  <super_flow>Aggregate.InfectionRate</super_flow>
  <equation>EXTERNAL (0) + (${NAME$.NeighboursAvg*0) + (${NAME$.Random*0) </equation>
  <parameter>${NAME$.Susceptible}</parameter>
  <parameter>${NAME$.Infectivity}</parameter>
  <parameter>${NAME$.NeighboursAvg}</parameter>
  <parameter>Aggregate.TimeStep</parameter>
  <parameter>${NAME$.Random}</parameter>
</flow>

```

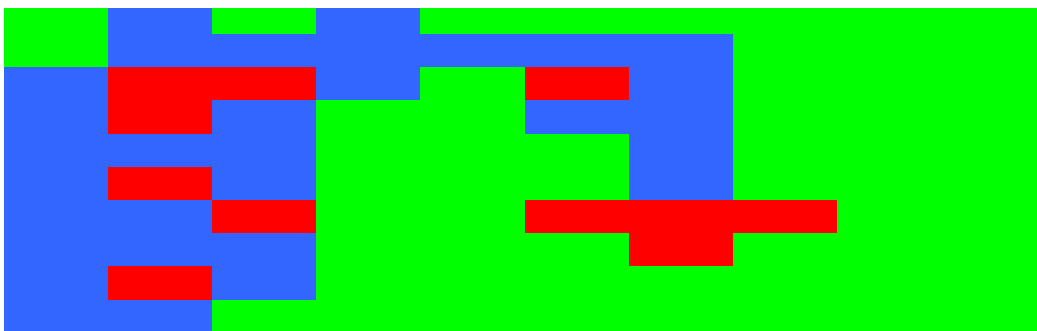


FIGURE 7 Information exchange between agents and the grid society structure

CASE STUDY: SIR MODEL

To illustrate how the system operates, a well-known case – the SIR Model - is selected. This has been widely modeled using both SD and agent-based methods. For this solution, the aggregate and agent components are shown in Figure 8. At an aggregate level the population is divided into three stocks: susceptible (S), Infected (I) and Recovered (R). The infection and recovery rates determine the rate of flow between these stocks. However, unlike the usual SD approach, these rate equations simply flow aggregators, and are determined by what happens at each individual agent level.

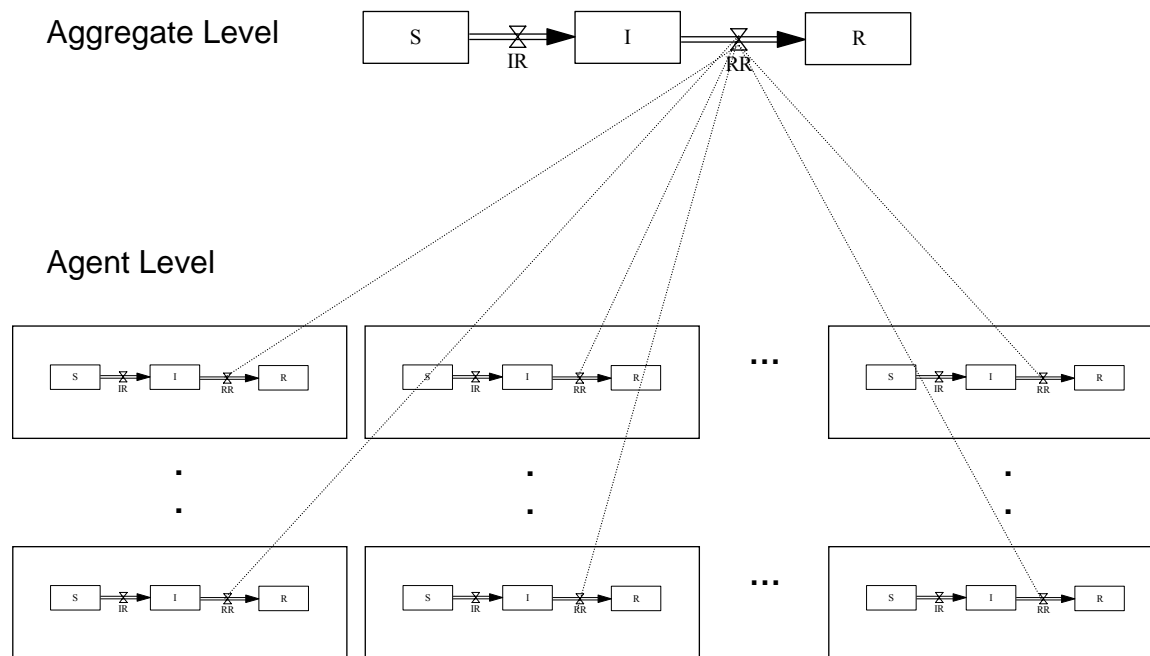


FIGURE 8 Two-level model of the SIR phenomenon

At each agent level, the SIR structure is employed, however in this case, the sum of all states must be 1 (i.e. an individual agent can only be one of S, I or R at any one time), and the rates are not continuous, so that states change at one point in time in a switching-type action. The recovery rate is simply a pipeline delay based on the infection rate (i.e. recovery follows infection after a certain number of time has elapsed). The infection rate is determined by: (1) the proportion of neighbours that are already infected and (2) the infectivity of the agent. An overall infection probability is calculated, and a [0,1] random number generated in order to decide whether an agent has been infected. If this happens, their state changes accordingly.

At a technical level, the complete model can be specified using mathematical equations, with a minimum of actual code. Underlying the model is a grid-based social network, where each

agent occupies a cell on the grid, and is influenced by its neighbours, a structure that has parallels with a cellular automata model. Sample results from a simulation are now shown.

Figure 9 shows the overall aggregate behaviour of the agents. There are 100 in total, and the stock of susceptible starts at 95, and depletes as the infection spreads. As the recover rates pick up, the spread of the infection slows, and an equilibrium is reached.

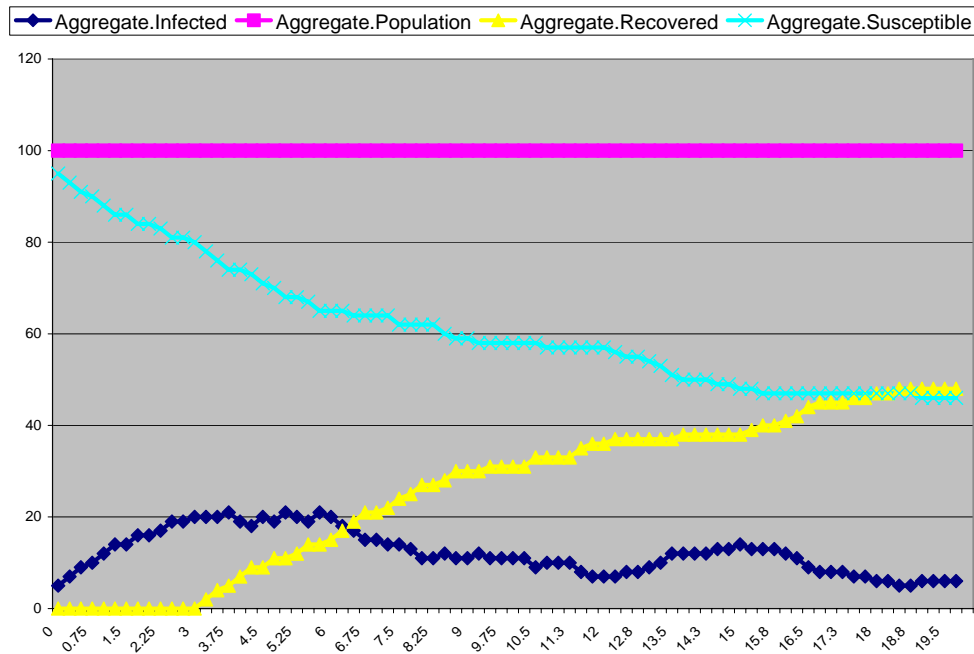


FIGURE 9 Aggregation of results by SIR categories

A more detailed analysis can also be viewed based on the overall state of the “grid” as time progresses. Figure 10 shows how the agent states have changed over a time interval of 5. Each rectangle represented 100 agents, and green maps onto susceptible, red is infected and blue recovered.

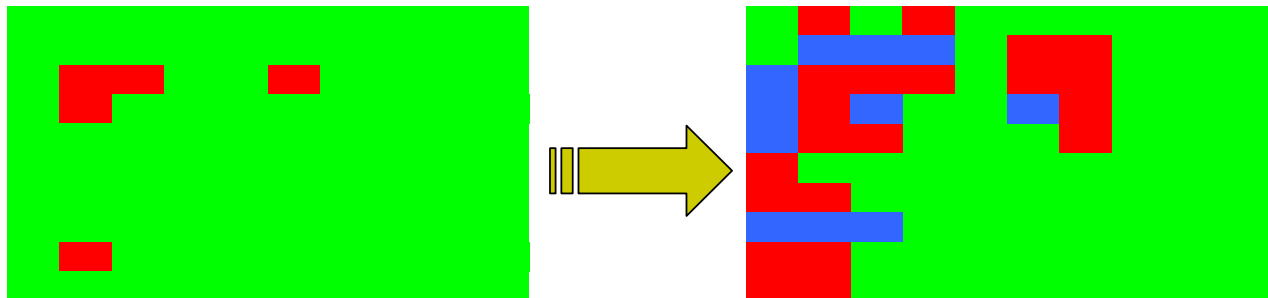


FIGURE 10 Change in agent states from time 0 to 5

Finally, Figure 11 shows a detailed trace of how individual states change over time. The time axis is vertical, and ranges from 0 through to 20. Each row across the diagram represents an agent's state at a particular point in time, and based on this it can be observed when an individual agent changed from one state to another.

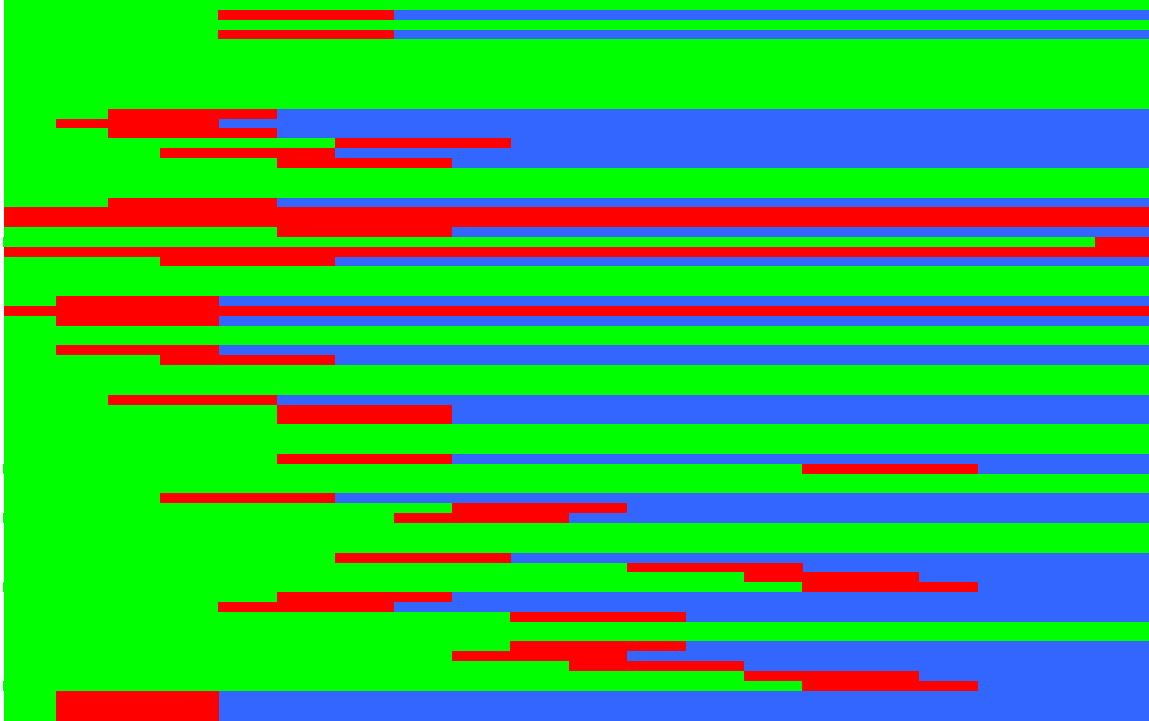


FIGURE 11 Mapping of agent state changes over time

CONCLUSIONS

This paper has presented an approach and a simulation system that can model agent-based systems using System Dynamics. There are a number of advantages to this, including:

- *Building on existing knowledge.* Models built using this approach have access to the rich body of knowledge within the field, including a wide variety of models that capture dynamic decision making processes across a range of disciplines.
- *Scalability and Performance.* Given a compact and lightweight numerical solver, this approach is scalable and should be able to accommodate a high number of agents and calculate results speedily.

Future work will include building a graphical user-interface for the current system, and also constructing a high performance numerical solver that will have the capability to simulate large populations of individual agents.

REFERENCES

- Akkermans, H. 2001. "Emergent Supply Networks: System Dynamics Simulation of Adaptive Supply Agents." *Proceedings of the 24th Hawaii International Conference on Systems Sciences*, ISSN 0-7695-0981-9/01.
- Borshchev A. and A. Filippov. 2004. "From System Dynamics and Discrete Event to Practical Agent Based Modeling: Reasons, Techniques, Tools." *Proceedings of the 22nd International Conference of the System Dynamics Society*. Oxford, England. 2004.
- Forrester, J.W. 1958. "Industrial Dynamics: a major breakthrough for decision makers." *Harvard Business Review*. 36, (4), 37-66.
- Forrester, J.W. 1961. *Industrial Dynamics*. MIT Press, Cambridge, MA.
- Forrester, J.W. 1969. *Urban Dynamics*. MIT Press, Cambridge, MA.
- Gilbert, N. and Troitzsch, K. 2005. *Simulation for the Social Scientist*, Open University Press.
- Lane, D.C. 2006. IFORS' Operational Research Hall of Fame: Jay Wright Forrester. *Intl. Trans. In Op. Res.* 13 (2006) 483-492.
- North, M.J., Collier, N.T., and Vos, J.R. 2006. "Experiences Creating Three Implementations of the Repast Agent Modeling Toolkit." *ACM Transactions on Modeling and Computer Simulation*. Vol.16, No.1, pp 1-25.
- Sterman, J. 2000. *Business Dynamics. Systems Thinking and Modeling for a Complex World*. McGraw Hill Higher Education