A Simulator for Continuous Agent-Based Modelling[1]

Jim Duggan,
Department of Information Technology,
Faculty of Engineering,
National University of Ireland, Galway.

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 a case study based on market dynamics. The overall benefits of the approach are summarised, and future work discussed.

1. Introduction

The term agent can be interpreted in a number of different ways, depending on the field of study. In computer science, and more specifically, distributed artificial intelligence, an agent is defined as a "computer system that is situated in some environment, and that is capable of autonomous action in this environment in order to meet its design objectives" Wooldridge (2002). This study of agency relates to the design of automated programs that can replicate human decision making in an intelligent manner, and so increase business returns and efficiency. The dominant view in the computer science literature concerns designing software agents to solve specific problems such as distributed rational decision making (Sandholm 2000), real time planning and scheduling (Parunak 2000), and computer supported cooperative work (Ellis and Wainer 2000).

In economics, a term often used is a rational agent, which "always acts in such a way as to have the maximum expectation of success given the information available" (Russell 1995). In this context, consumers in a marketplace would be examples of rational agents, as they seek to optimise their returns based on the availability of resources and information. This idea of a rational agent is also closer to the dictionary definition (Collins 1994), which defines an agent as "a person or thing that acts or has the power to act," thus indicating that the entity has a degree of autonomy, the ability to process information, and the power to take action. In social systems, management may be viewed as agents, as they are empowered to act in the best interests of an organisation, and their 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). The idea of modelling agency, in terms of simulating the behaviour of those with the "power to act," and observing the influence of their actions on the overall system behaviour, has been central to system dynamics from its inception.

[1] The support of *NUI, Galway's Millenium Fund* is gratefully acknowedged

While Forrester's (1961) earlier work modeled agents taking action, a feature of these supply chain models were, when compared to modern agent-based simulation models such as RePast (North et al. 2006) and AnyLogic (Borschev and Filippov 2004), that the number of agents were relatively small. In recent years there have been exceptions to this, for example, in the domain of corporate networks dynamics, Akkermans (2001) presents a system dynamics model of 100 agents. Another important distinction is that agent-based modeling that is not based on SD takes its root from computer science, and is founded on the idea of a software object, namely, a specifically written software program that encapsulates an agent's state and behaviour. For example, Gilbert and Troitzsch (2005) comment that "a natural way of programming agents is to use an object-oriented programming language," and it is not surprising that ABM toolkits provide an object-oriented framework where modelers must rely on a good knowledge of programming in order to model agents and control their interactions. These toolkits and approaches also operate using discrete-event simulation as the underlying mechanism for controlling the passage of time.

In summary, approaches to agent-based simulation can be classified under two broad headings. The first, heavily influenced by computer science, regard an agent as an extension of a software object, and modelers must employ programming knowledge in order to construct simulation models. An exception is AnyLogic, but the vendors recommend that in order to maximise performance, the use of differential equations should be minimised[2]. Furthermore, while the approach presented here can be implemented in AnyLogic, to do so requires the use of Java to perform the necessary flow aggregations and society network configuration.

The second approach to agent-based simulation employs system dynamics 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 these shortcomings by providing an approach and technology that allows large scale agent models to be built, based entirely on sets of equations, and does not require programming expertise to do so.

2. System Design

Figure 1 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

[2] http://www.xjtek.com/support/kb/search.xml?keyword=performance (See KB155)

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. While agents will normally share similar states, they may differ greatly in terms of their decision rules, or the manner in which they process available information.

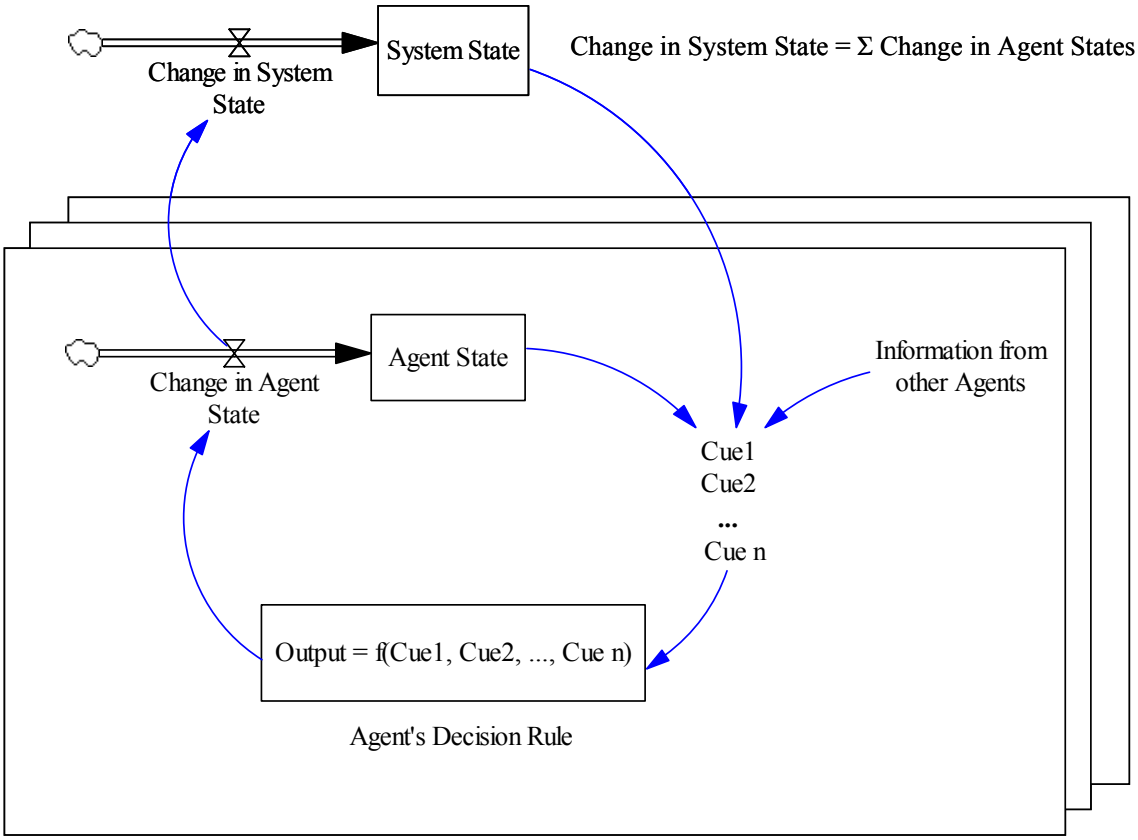


Figure 1: Conceptual Design for Continuous Agent-Based Modelling

In order to extend a convention system dynamics solver so that it can accommodate the requirements of this two-level conceptual design, three key features must be added.

- A flow aggregator, which allows a high level flow to be composed of many lower-level flows, and thereby enable decisions made at a detailed level to be captured at higher levels. For example, if the lower levels modeled individual agents choosing between two different telecom suppliers, the higher level would aggregate all of these individual decisions to present the bigger picture of the relative strengths of each installed base.
- A top-down information mapping, which allows models at a lower level of detail to have access to information cues that operate at the higher levels. This would allow individual agents to be influenced by market information, for example, the overall popularity of a product.

- A social network structure, that allows models at a lower level (i.e. individual agents) to access information cues from neighbouring agents. In this way, agents can be influenced by their peers, and therefore interesting dynamics can emerge in a bottom-up manner, in contracts to the top-down information mapping.

The system architecture is captured in Figure 2. The major software components are:

- Continuous Agent-Based Modeling (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 can be 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, which 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.

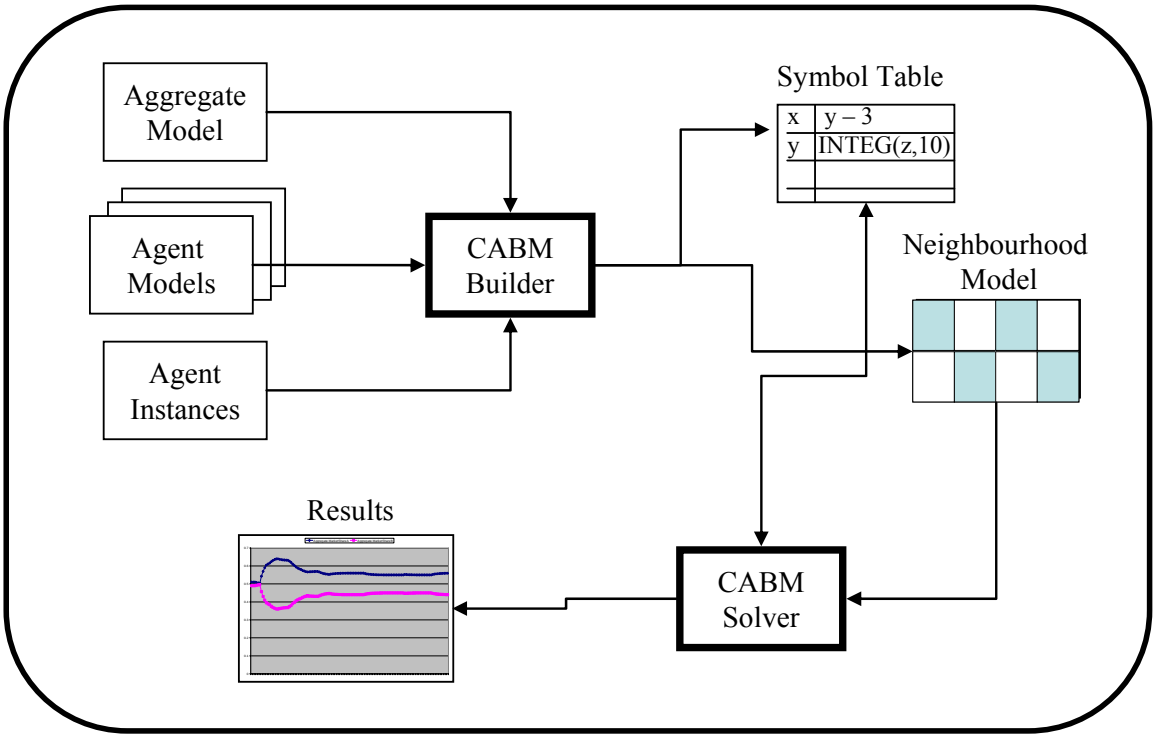


Figure 2: Continuous Agent-Based Modeling Architecture

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

- The *aggregate model*, which corresponds to the "System State" element of Figure 1. 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. For example, if the object of interest was the Beer Game model (Sterman 2000), two separate agent models could be constructed. The first could take the supply line into account, the second model could ignore the supply line.
- 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_aggregrate>" as part of it definition (see figure 3). 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.InstalledBaseA</name>
      <is_aggregate>true</is_aggregate>
      <init>0.0</init>
      <inflow>Aggregate.TotalDefectRateFromB</inflow>
      <outflow>Aggregate.TotalDefectRateFromA</outflow>
</stock>

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

Figure 3. 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 (see figure 4). In this example, the string "$NAME$" will be replaced by the specified agent name when the model created.

```
<flow>
      <name>$NAME$.DefectFromA</name>
      <is_subflow>true</is_subflow>
      <super_flow>Aggregate.TotalDefectRateFromA</super_flow>
</flow>
```

Figure 4: Defining Flows at an agent level

The second mapping concerns connections from the aggregate level to the detailed level. These are simpler to implement, as it involves assigning a single variable from a higher-level model to a corresponding variable in an agent model. An example of this is shown in figure 5, where the agent variable $NAME$.LoadA is assigned the value Aggregate.LoadA.

```
<auxiliary>
      <name>$NAME$.LoadA</name>
      <equation>Aggregate.LoadA</equation>
</auxiliary>
```

Figure 5. Mapping higher level variables to lower agent levels

The final mapping relates to agent-to-agent communications, where and agent uses information from other agents in order to arrive at a decision. In order to facilitate this, the agents are modeled as a society in a grid-like structure. In figure 6, a society of 100 agents are shown, where those in blue (56) are associated with company A, whereas the red squares are customer's of company B (44). The cells are assigned randomly (although there is an option for clustering), and it is possible to base an agent's decision on the state of the cells in its neighbourhood. For example, the purpose-built function *NEIGHBOURHOOD_AVERAGE()* – see figure 7 – 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. For example, in the case study presented in the following section, this is used each agent's to estimate the popularity of a given supplier.

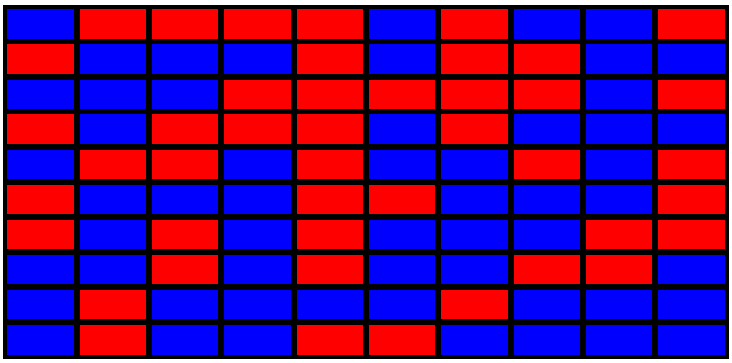


Figure 6. An agent society

```
<auxiliary>
      <name>$NAME$.MarketShareA</name>
      <equation>NEIGHBOURHOOD_AVERAGE($NAME$,$NAME$.ChooseSupplierA)
      </equation>
</auxiliary>
```

Figure 7. An agent calculating the neighbourhood average

3. Case Study

To validate the system, a market dynamics model is presented (figure 8). At an aggregate level, the model has two stocks, each representing the installed base for a different company. In this two-company marketplace, the stock of all customers is constant, and customers move from one company to another, depending on what state the system is in.

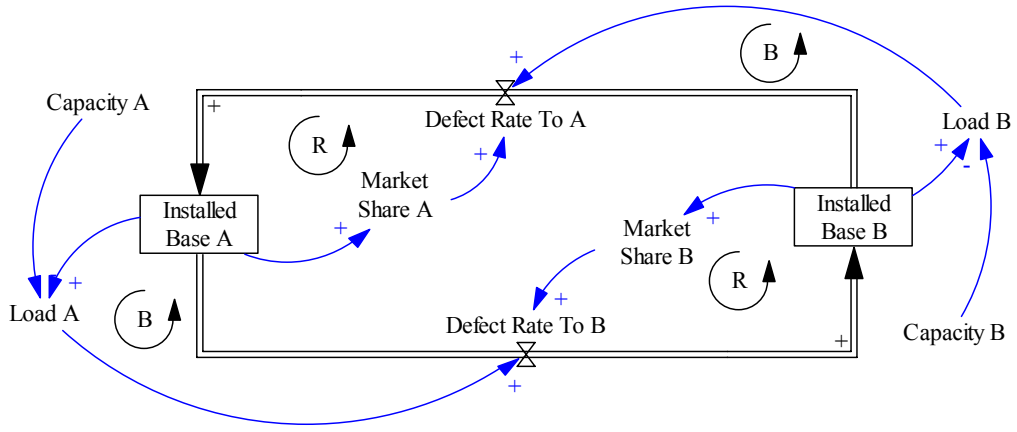


Figure 8: Aggregate market dynamics model

For each company, there are two feedbacks that influence attraction and retention of customers. The first is a positive feedback, which could be termed "success to the successful", where an increased market share results in an increased number of customers. The second balancing feedback, could be called "more customers, higher load, less customers", where the capacity strain resulting from increased customers leads to the loss of customers in the long run. For such a model, if the positive feedback were to operate on its own, the system should exhibit path dependence, and whichever company is ahead at the start would go on to dominate. However, if the more quality-oriented feedback loop was solely in operation, the model should seek an equilibrium, whereby each company would attract customers so that they operate close to capacity.

The aggregate stocks are formulated in equations (1) and (3). The initial values for these stocks cannot be specified in advance, and depend on the number of agents who have initially opted for company A or company B. The system contains a summation utility which will gather this information before the simulation commences (2) and (4).

(1) Installed Base A = INTEG(Defect Rate To A – Defect Rate To B, InitA)
(2) InitA = Σ Initial States for Stock Choose Supplier A for all Agents
(3) Installed Base A = INTEG(Defect Rate To B – Defect Rate To A, InitB)
(4) InitA = Σ Initial States for Stock Choose Supplier A for all Agents

The two high-level flows are described in equations (5) and (6), and these formulations are aggregations of what occurs at the individual agent-level of the model.

(5) Defect Rate to A = Σ Defect Rate to A for all Agents
(6) Defect Rate to A = Σ Defect Rate to A for all Agents

Based on the two stocks, the market share for each company can be calculated (7) and (8). Each company's capacity (units are people) is shown in (9) and (10), and a step function is used to change these values half way through the simulation, in order to observe the resulting dynamics. The loading factor both company's is a ratio of their installed base and capacity (11) and (12).

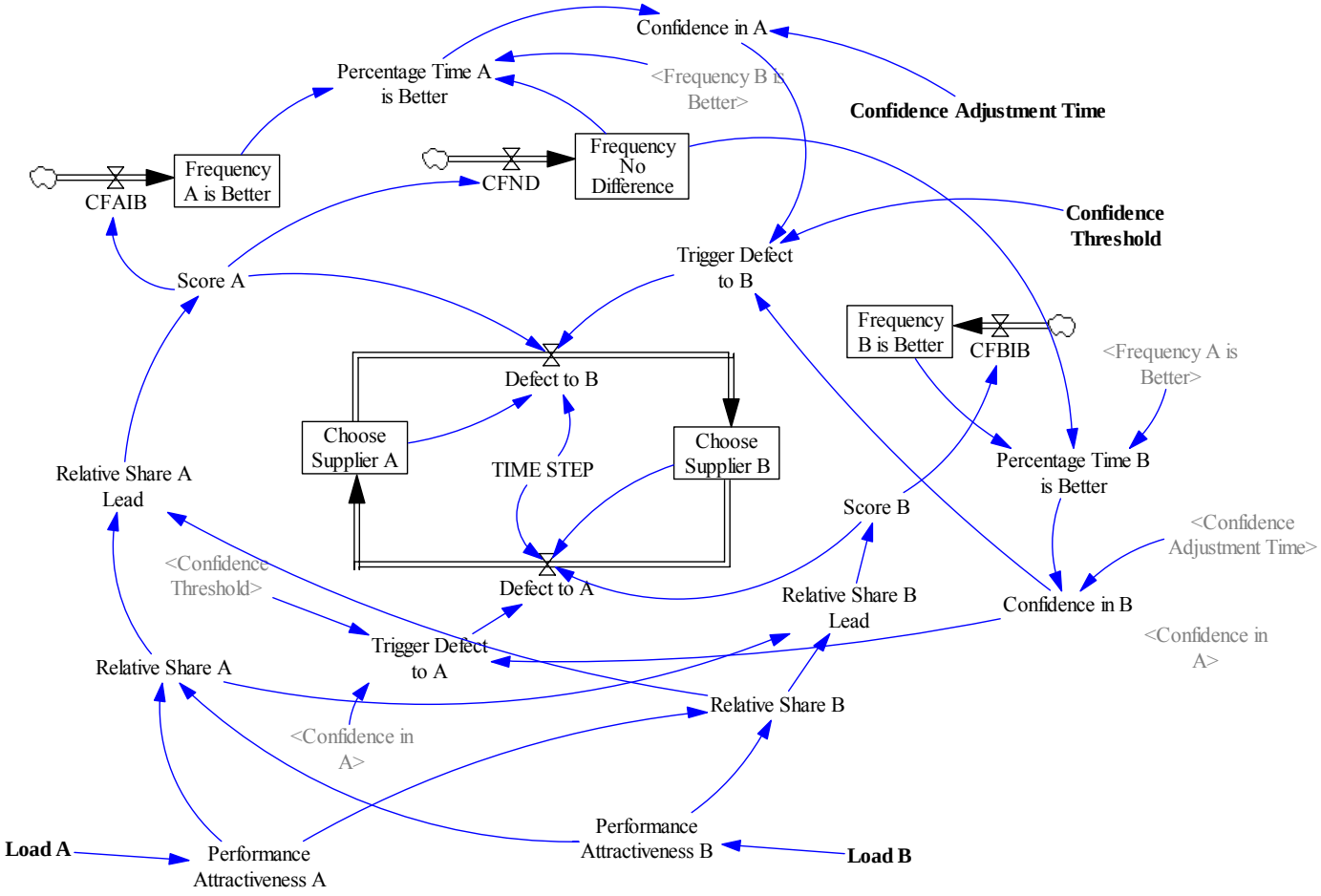
(7) Market Share A = Installed Base A / (Installed Base A + Installed Base B)
(8) Market Share B = Installed Base B / (Installed Base A + Installed Base B)
(9) Capacity A = 40 + STEP(20,20)
(10) Capacity B = 60 - STEP(20,20)
(11) Load A = Installed Base A / Capacity A
(12) Load B = Installed Base B / Capacity B

In the case study, there are three different types of agent, which share common features, but differ in terms of the information they use to decide which company to subscribe to. These three agents are:

- The *Fact-Based Agent*, which uses the load-related information from equations (11) and (12) in order to decide which company to subscribe to.
- The *Fad-Based Global Agent*, which considers the market share information from equations (7) and (8) as part of their decision making process.
- The *Fad-Based Local Agent*, which interacts with its neighbours in order to discover their preferences, and uses this figure as an estimate for the market share of each company.

The stock and flow structure for the fact-based agent is shown in figure 9. Two stocks (13) and (14) are used to represent which supplier is subscribed to, and this structure mirrors the stock and flow structure at the aggregate level. The equations that govern the transition from one supplier to another are summarised in (15) and (16).

(13) Choose Supplier A = INTEG(Defect to A - Defect to B , 1)
(14) Choose Supplier B = INTEG(Defect to B - Defect to A , 0)



(15) Defect to A = IF THEN ELSE (Score B = -1, ((max (Choose Supplier B , Score B)) * Trigger Defect to A) / TIME STEP , 0)
(16) Defect to B = IF THEN ELSE (Score A = -1, ((max (Choose Supplier A , Score A)) * Trigger Defect to B) / TIME STEP , 0)

Figure 9. Stock and flow model for a Fact-Based Agent

Structurally both of these equations are similar, and their logic can be described as follows

- If B's score is -1 (i.e. B has lost the most recent comparison), and B is currently winning, and if the agent's confidence in A exceeds their confidence in B by more than the threshold, then a defection from A to B occurs. (15)

- If A's score is -1 (i.e. A has lost the most recent comparison), and A is currently winning, and if the agent's confidence in B exceeds their confidence in A by more than the threshold, then a defection from B to A occurs. (16)

Where defections occur, they do so in discrete jumps, as the rate in question is divided by the time step in order to achieve a pulse-like effect. This is a recognition of how customer's change suppliers in the real world, with an "all or nothing" switching action. When the results at the agent level are eventually aggregated up, these two equations, summed over the entire population, will give the installed base variables a discrete appearance, and will also mean that the aggregate state values are always whole numbers. The logic for each trigger is contained in equations (17) and (18). The basis for each agent deciding to make a change is the relative difference in their confidence in each supplier. The confidence (20) is a smoothed average of the percentage time that A is better that B, and the confidence adjustment time (22) is a constant that reflects how quickly the confidence changes in relation to the target variable. In later experiments, equations (19) and (22) will be varied depending on the profile of each cohort of agent, for example, longer adjustment times will be given to less volatile customers, who may wait longer in order to see a trend before making a decision to switch suppliers.

(17) Trigger Defect to A = IF THEN ELSE (Confidence in A - Confidence in B > Confidence Threshold, 1, 0)
(18) Trigger Defect to B = IF THEN ELSE (Confidence in B - Confidence in A > Confidence Threshold, 1, 0)
(19) Confidence Threshold = Specified at runtime depending on Agent Profile
(20) Confidence in A = SMOOTHI (Percentage Time A is Better , Confidence Adjustment Time, 0.5)
(21) Confidence in B = SMOOTHI (Percentage Time B is Better , Confidence Adjustment Time, 0.5)
(22) Confidence Adjustment Time = Specified at runtime depending on Agent Profile

Equations (23) through (30) are used to calculate the percentage of time each supplier scores better than the other.

(23) Percentage Time A is Better = zidz (Frequency A is Better , (Frequency A is Better+ Frequency B is Better + Frequency No Difference))
(24) Percentage Time B is Better = zidz (Frequency B is Better , (Frequency A is Better+ Frequency B is Better + Frequency No Difference))
(25) Frequency A is Better = INTEG(CFAIB , 0)
(26) Frequency B is Better = INTEG(CFBIB , 0)
(27) Frequency No Difference = INTEG(CFND , 0)
(28) CFAIB = IF THEN ELSE (Score A = 1, 1, 0)
(29) CFBIB = IF THEN ELSE (Score B = 1, 1, 0)
(30) CFND = IF THEN ELSE (Score A = 0, 1, 0)

A score is produced for each supplier – equations (31) and (32) - over each time period. If a supplier is better than its competitor, it receives a score of 1. If both suppliers score equally, their scores are 0, and if a supplier loses, its score is -1.

(31) Score A = WITH LOOKUP(Relative Share A Lead , ([(-1,-1)-(1,1)],(-1,-1),(-0.5,-1),(-0.25,-1),(-0.1,-1),(-0.05,-1),(-0.01,-1),(0.005,1),(0,0),(0.005,1),(0.01,1),(0.1,1),(0.4,1),(0.5,1),(0.6,1),(0.7,1),(0.8,1),(0.9,1),(1,1))

(32) Score B = WITH LOOKUP(Relative Share B Lead , ([(-1,-1)-(1,1)],(-1,-1),(-0.5,-1),(-0.25,-1),(-0.1,-1),(-0.05,-1),(-0.01,-1),(0.005,1),(0,0),(0.005,1),(0.01,1),(0.1,1),(0.4,1),(0.5,1),(0.6,1),(0.7,1),(0.8,1),(0.9,1),(1,1))

The scores for A and B are based on the relative share lead (33-34) of each supplier, which amount a supplier leads its competitor by. The relative share (35-36) of each supplier is based on their performance attractiveness (37-38), which in turn is a function of the load on each supplier (11-12).

(33) Relative Share A Lead = Relative Share A - Relative Share B

(34) Relative Share B Lead = Relative Share B - Relative Share A

(35) Relative Share A = 0.5 + (Performance Attractiveness A - Performance Attractiveness B)

(36) Relative Share B = 0.5 + (Performance Attractiveness B - Performance Attractiveness A)

(37) Performance Attractiveness A = WITH LOOKUP(Load A , ([(0,0)-(6,1)],(0,0.5),(0.5,0.5),(1,0.5),(2,0.3),(3,0.1),(4,0.1),(5,0.1),(6,0.1))

(38) Performance Attractiveness B = WITH LOOKUP(Load B , ([(0,0)-(6,1)],(0,0.5),(0.5,0.5),(1,0.5),(2,0.3),(3,0.1),(4,0.1),(5,0.1),(6,0.1))

To further explain this final set of equations, consider the following scenario. We will assume the following values for equations (11) and (12).

Load A = .75
Load B = 1.25

Based on these values, equations (37) and (38) will evaluate to:

Performance Attractiveness A = 0.5
Performance Attractiveness B = 0.3

Following that, equations (35) and (36) will take on the values:

Relative Share A = 0.5 + (0.5-0.3) = 0.8
Relative Share B = 0.5 + (0.3-0.5) = 0.2

Working these through equations (33), (34), (31) and (32) will result in a score of 1 for A (a win), and a score of -1 for B (a loss).Which means that for a *Fact-Based Agent,* B is

now a more attractive proposition that A, and that if this situation remains over time, eventually the agent will switch from A to B. The structures for the remaining agents are mostly similar, with the following exceptions:

- For the *Fad-Based Global Agent,* the information cue used to determine attractiveness is aggregate market share, as specified in equations (7) and (8).
- For the *Fad-Based Local Agent,* the market share for both A and B is a neighbourhood average from each of the surrounding agents, based the equation (13) for A, and equation (14) for B.

Following on from this formal description of the model, experimental results are presented in the next section.

4. Experimentation

To validate the approach, a set of experiments are presented based on the models developed in the previous section. The idea is to illustrate how a heterogeneous population can be constructed using these models. Two main experiments are carried out.

- The first is designed to test the reinforcing feedback loop - *success to the successful* – by limiting the population to fad-based agents, and ensuring that one supplier has an initial early lead. The expected result here would be a "winner-takes-all" scenario.
- The second experiment creates a mixed population of fact and fad-based agents, and introduces a capacity variation in order to test whether the negative feedback loop - *more customers, higher load, less customers* - can become the dominant feedback structure in the model.

The experimental consumer population is segmented into a number of cohorts – based on age grouping – and the values of key parameters are arbitrarily adjusted to reflect how different cohorts decide as to their supplier preference. Within each cohort, there can be both fact and fad agents, and the proportions within each cohort can be varied for each simulation run. These parameters are *confidence threshold* (19) and *confidence adjustment time* (22). More volatile decision makers will have lower values, and in a sense these values model the patience of consumers. The different cohorts, and their adjustment times are listed in table 1. Because a core idea of agent-based modeling is to capture a range of behaviours, the values for their adjustment times are taken from a uniform distribution, so as to maximise the differences between agents.

	Fact Agents		**Fad Agents**	
Cohort	**CAT (22)**	**Threshold (19)**	**CAT (22)**	**Threshold (19)**
Seniors	[40.0-60.0]	0.50	[20.0-60.0]	0.50
Boomers	[20.0-40.0]	0.35	[10.0-30.0]	0.35
Generation X	[10.0-25.0]	0.25	[5.0-20.0]	0.25
Generation Y	[2.0-10.0]	0.15	[1.0-5.0]	0.05

Table 1: Adjustment times and thresholds for agents

Experiment 1: Simulating FAD Agents

For this experiment four different simulations were performed based on simulating a (56-44) split on initial market share. The summary results are shown in figure 10, where the market share of company A is charted.

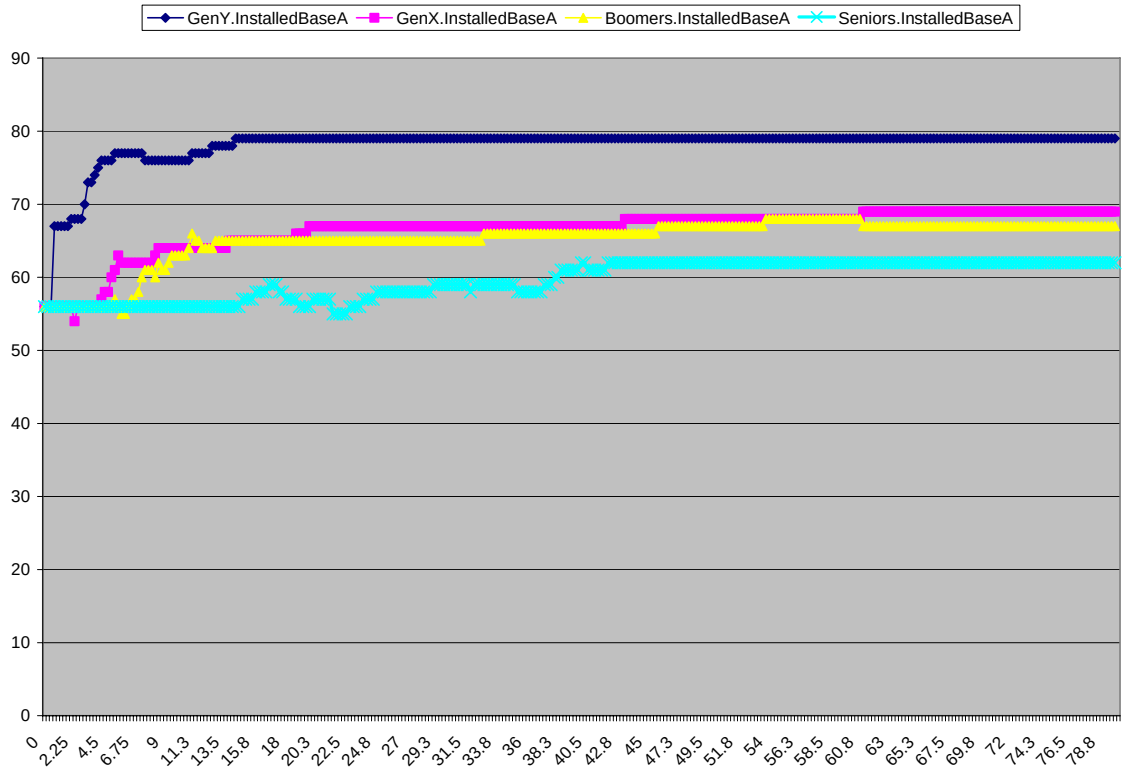


Figure 10. Results from simulations where all agents are fad

On examining the results, one might have expected more dominance for company A, because the positive feedback dynamic, which is the only one present, would suggest that a situation of complete dominance would emerge over time. However, a more detailed look at the underlying society structure explains why company A only gets so far. For example, the society structure for the first simulation, generation-Y, is shown in figure 11.

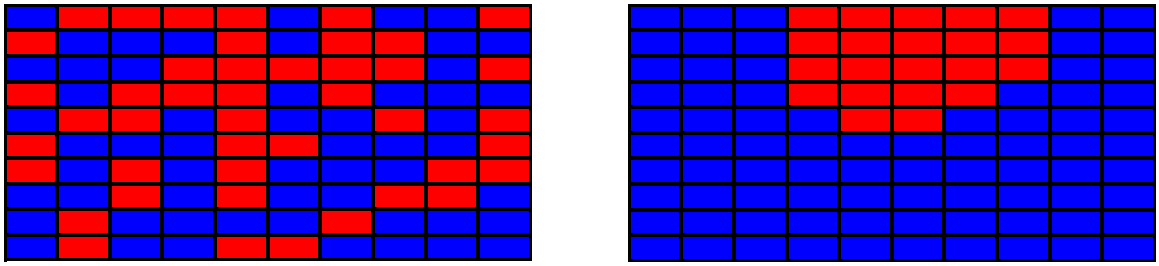


Figure 11. Initial and final states for generation-Y simulation

The initial state is randomly assigned, but overall there is a majority of agents that subscribe to company A. The word of mouth mechanism is based on the proportion of an agent's neighbours that use a given product. Over time, while the blues win out, they can only get so far because a cluster of reds together mean that no more reds will defect to blue, as the majority of each individual red agent's neighbours are red.

A second feature of the results worth commenting is that each successive cohort locks in at lower market share. For example, generation-Y locks in at around 80%, whereas the Seniors lock in at just over 60%. The reason for this has to do with the value for confidence threshold. A large confidence threshold, which is what the seniors have, means that the agent tolerates a greater difference between the relative scores of the companies, and so individual agents are slower to defect.

Experiment 2: Simulating a Mixed Population

This experiment explores the results for a mixed population of fact and fad agents. The expected result here is that the negative feedback loop will dominate, as the issue of quality of service will arise. The agent breakdown is displayed in table 2, and at an aggregate level company A will start with an installed base of 53.

Cohort	Fact Agents		Fad Agents	
	Company A	**Company B**	**Company A**	**Company B**
Seniors	9	7	2	2
Boomers	9	9	6	6
Generation X	7	5	9	9
Generation Y	2	2	9	7
Totals	27	23	26	24

Table 2. Initial conditions for the mixed simulation run

For this simulation, because we now have fact-based agents in the population, the issue of supplier capacity, detailed earlier in equations (9) and (10), now becomes important. Initially, B's capacity is greater, but half way through the simulation the situation is turned around, where B loses capacity and A gains. Figure 12 captures the simulation result. Here we can see that A loses its initial advantage to B, and that over the first half of the simulation, B peaks at a dominant position. However, the switch in capacity has the effect of turning the tables, and customers defect back to A so that by the end of the simulation, A is once more in a dominant position. An important observation is that while this figure shows the aggregate behaviour, it does not indicate what is happening at a detailed level. For example, in the early phase of the simulation, several agents defect from B to A (fad agents), but these are outnumbered by the number of fact agents defecting in the opposite direction. A detailed view of individual agent states is shown in figure 13, and it shows how agents move from A to B and back to A again, before the system settles at equilibrium.

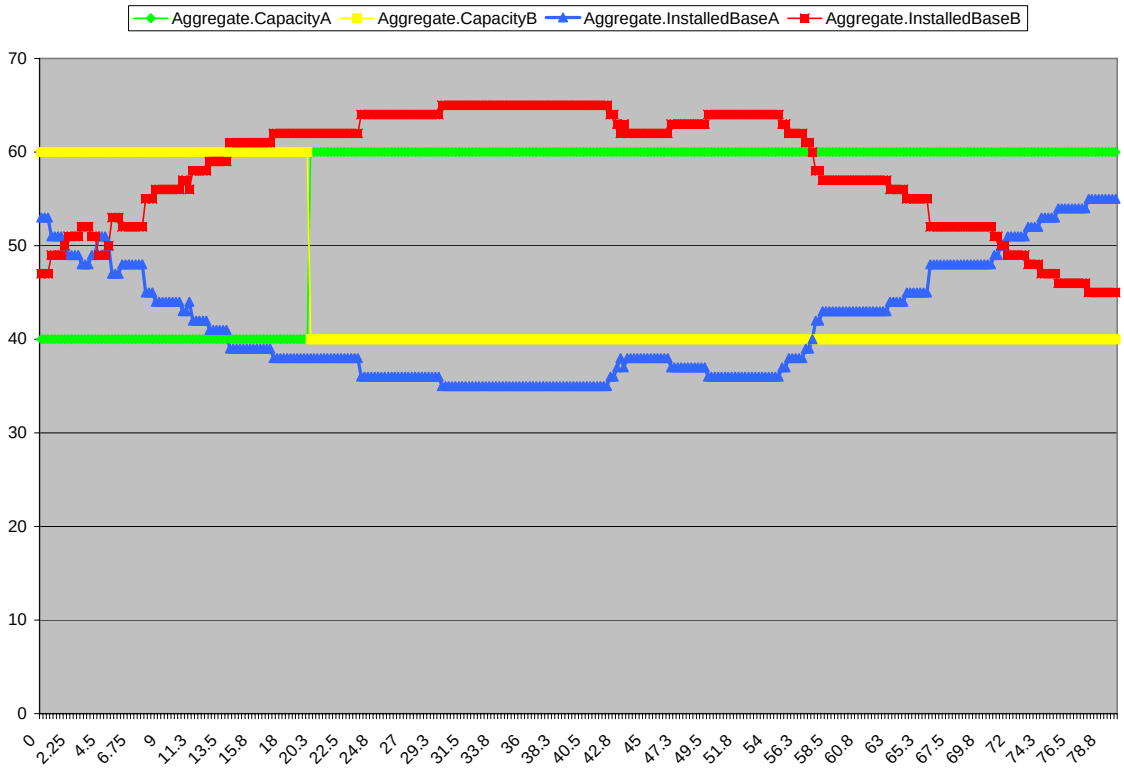


Figure 12. Simulation results for mixed population

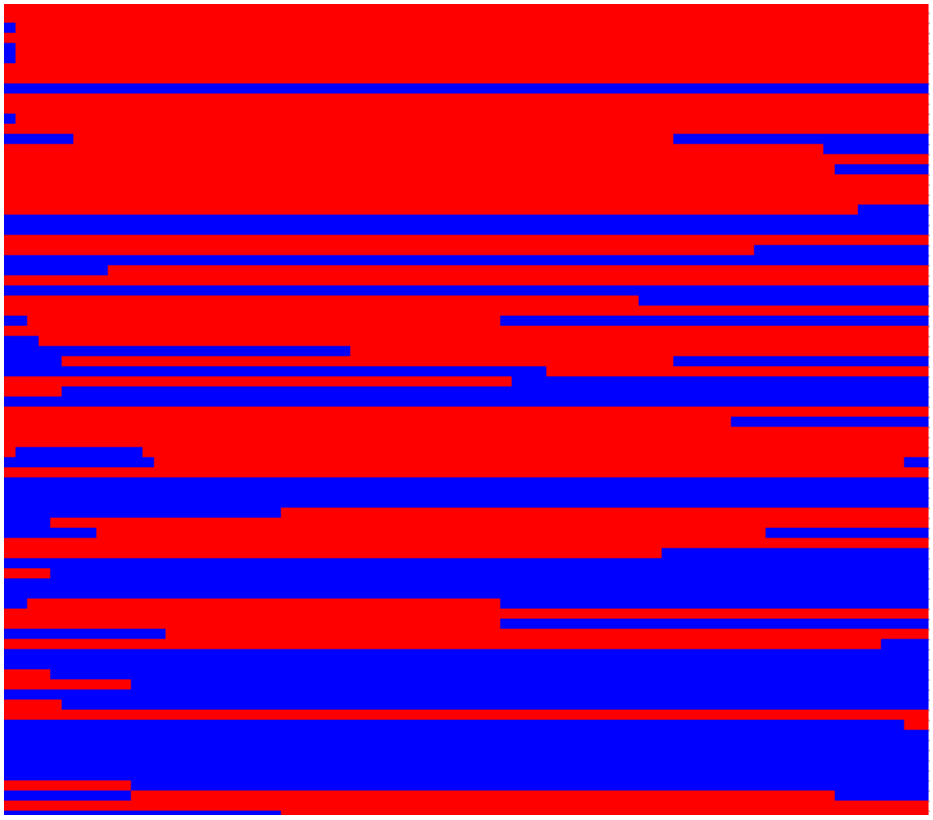


Figure 13. Individual agent preferences over time

5. Conclusion

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.

- *Diversity.* There is no limit to the amount of diversity captured in an agent, once its behaviour can be captured as a stock and flow model.

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.

6. 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.

Collins 1994. "Collins English Dictionary." HarperCollins Publishers, Glasgow, UK.

Ellis, C. and Wainer, J. 2000. "Groupware and Computer Supported Cooperative Work." From the book Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence, Edited by Gerhard Weiss, MIT Press, Cambridge Mass., USA.

Forrester, J.W. 1969. Urban Dynamics. MIT Press, Cambridge, MA.

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.

Parunak, H. Van Dyke. 2000. "Industrial and Practical Applications of DAI." From the book Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence, Edited by Gerhard Weiss, MIT Press, Cambridge Mass., USA.

Sterman, J. 2000. *Business Dynamics. Systems Thinking and Modeling for a Complex World.* McGraw Hill Higher Education.

Wooldridge, M. 2002. An Introduction to MultiAgent Systems. Wiley Publishers, Chichester, England.