1-1-2013

# Benefits of generalised microsimulation

Daniel Keep
*University of Wollongong*, drk02@uowmail.edu.au

Ian Piper
*University of Wollongong*, ian@uow.edu.au

Anthony Green
*University of Wollongong*

# Benefits of generalised microsimulation


## Abstract

This paper states our position regarding the desirable properties of a discrete simulation environment and details our response to this. Following a brief introductory examination of the pre-existing art in microsimulation modelling, we describe our approach and detail its structure and use.



## Keywords

microsimulation, benefits, generalised


## Disciplines

Engineering | Science and Technology Studies


## Publication Details

Keep, D., Piper, I. & Green, A. (2013). Benefits of generalised microsimulation. In Q. Bai, T. Ito, M. Zhang, F. Ren & X. Tang (Eds.), The International Workshop on Smart Simulation and Modelling for Complex Systems (SSMCS 2013) (pp. 13-24). China: IJCAI.

# Benefits of Generalised Microsimulation

Daniel Keep, Ian Piper, and Anthony Green

University of Wollongong
ian@uow.edu.au

**Abstract.** This paper states our position regarding the desirable properties of a discrete simulation environment and details our response to this. Following a brief introductory examination of the pre-existing art in microsimulation modelling, we describe our approach and detail its structure and use.

## 1 Introduction

One of the notable techniques in discrete event modelling is microsimulation modelling (MSM). Unfortunately, MSM is typically discussed in terms of specific application domains, rather than as a general technique in itself. We believe that this has led to the potential of MSM, as a tool for complex multi-domain simulations, to be largely overlooked.

Although there has been some work towards such generalisation [17], its focus has been largely on tools for the development of simulation programs. We are interested in building a framework that allows for the easy reuse and composition of simulations through the use of components. In addition, it is our contention that it should be possible for non-trivial simulations to be constructed by non-programmers.

We are also interested in combining the best aspects of various simulation techniques, using our MSM constructs as a common base. This, we believe, facilitates the creation of integrated systems for modelling of real-world problems involving multiple phenomena and processes.

We believe these aims are feasible; the existence and demonstrated utility of our MSM framework, simulation modules and associated tools confirm this position, as well as providing a useful base for further research and development.

## 2 Microsimulation Modelling

Microsimulation modelling was first described by Orcutt in 1957. In MSM, the primary modelling construct is the individual [11]. The population is modelled either by a representative sample, or by a complete census. In both cases, each member of the population has its own state and behaviour. This is in contrast to the standard modelling techniques of the day which aggregated the state and behaviour of the individuals which made up the population.

MSM techniques are applicable to a wide range of problem domains ranging from epidemiology [4] to transportation [20]. However, there appears to be an unfortunate level of "ghettoisation", in that references to MSM are almost always made in the context of a particular area of application. There are many examples of *transport* microsimulation, *social* microsimulation, or *economic* microsimulation, but comparatively few examples of *general* microsimulation.

### 2.1 On Agents and MSM

Spielauer notes that agent-based simulation can be considered a kind of dynamic MSM, but generally is not [16]. He states that both techniques developed almost entirely in isolation from one another.

Crooks and Heppenstall state that the crucial difference between an agent-based approach and MSM is that MSM does not model the impact individuals can have on the model's policy, nor does it model interactions between individuals [6]. This is not a hard-and-fast rule; existing work shows how elements of agent-based modelling can be introduced into an MSM context [21, 9].

It is interesting to note that Stroud *et al.* refer to EpiSimS (which is arguably a dynamic microsimulation) as an agent-based simulation [18], whilst Chen refers to the same approach variously as "microsimulation", "agent-based simulation" and "agent-based microsimulation" [5]. In practice, there is much common ground between the two approaches.

### 2.2 Observations on the State of the Art

We are primarily interested in modelling the behaviour and response of individuals. Both MSM and Agent-Based Modelling (ABM) were developed to facilitate such modelling. In practice, MSMs are built for specific purposes. For example, TRANSIMS is a transport microsimulation [15]. Even if you are using a framework such as ModGen or UMDBS [14], the end result is a self-contained, stand-alone simulation program with a specific, well-defined purpose.

It is our belief that a greater emphasis on the development of generalised microsimulation technologies would be of great benefit, both in domains already served by existing tools and in domains as yet untapped. Although there is unquestionable value in building simulations for specific areas of interest, there is also a need for simulations which cover multiple problem domains. An example of this is the work of Perez *et al.* at the University of Wollongong in which the interaction between population changes and the transportation network are being analysed. Despite this being a natural problem to examine, a significant portion of their work has been spent on interfacing the two major simulations components [12]. This would also serve to combat the ghettoisation of MSM.

There appears to be a general perception that models which involve behaviour necessitate the use of "full-blown" ABM techniques. We believe that a considerable (and very useful) degree of expressivity can be achieved by introducing simple, limited aspects of ABM to MSM, rather than abandoning MSM in favour of ABM. Birkin and Wu would appear to share this belief, stating:

"...it is more constructive to view the relationship of ABM to MSM as one of complementarity rather than supremacy." [3]

### 2.3 Methods for Developing a New Simulation

There are three broad strategies for constructing a simulation. First, the simulation can be created from scratch. Secondly, the simulation can be built on a simulation framework. Thirdly, the simulation can be assembled using one or more simulation components.[1] Of these, we believe the third to be the most interesting and attractive.

Creating simulations from scratch naturally involves a significant amount of new design and coding effort. Some of this work can be saved by modifying or combining existing simulations. This still requires a significant amount of time and effort, in some cases potentially more than would be required to write it from scratch [12]. This is particularly true if combining simulations with substantially different data formats, interfaces, approaches to time and space, etc.

Frameworks significantly reduce this burden by implementing common pieces of functionality. If implementing an entirely novel model, this is probably the most attractive approach. However, most frameworks are designed to produce stand-alone, single-purpose simulations, which cannot be readily re-purposed or combined. Thus, there may still be significant barriers to combining disparate simulations, even within the same framework.

Our approach is to design the framework not just to implement commonly required support code, but to also ensure that multiple, independent processes can co-exist within the same simulation without requiring code changes at any level. Once a particular capacity has been implemented atop the framework, it is available for use alone or in combination with any other such components, without additional code. The desired traits of reusability and composability are inherent in this strategy, the value of which have been previously recognised, at the conceptual level, by Balci *et al.* [2]. This has been borne out in practice, with the non-programming members of our team able to construct new, complex simulations without the need for a programmer.

## 3 Our Work

The work of our group is focused on developing and applying a generalised approach to the construction of MSM simulations. This has resulted in the creation of the *Simulacron* MSM framework (and supporting code), a number of simulation modules, and associated tools, as well as the design and construction of a number of simulations which make use of the framework, modules and tools. Further details on these can be found in [10]. The guiding principle behind the design of this software has been "keep it simple; if you can't, keep it general."

[1] Ideally, these components would be pre-existing, although additional ones could be written with no more effort than required in the second approach.

### 3.1 Simulacron Framework

Conceptually, the framework's most important task is defining the set of fundamental abstractions upon which all simulation components must be built. Without this set of shared abstractions, independent simulation components cannot easily cooperate.

Simulacron implements discrete time MSM, meaning that the simulation is advanced by one or more "ticks". Ticks can be of uniform or varying length, and any integer number of milliseconds long.

Simulacron depends on two key types of entities for representing the world; one for locations and one for individuals. Locations are represented as "cells", which have no inherent properties other than a unique identifier. Individuals are "peeps", which are similarly minimal, knowing only their own identifier and which cell they are currently located in.

The data associated with each simulation process is attached to cells and peeps via "fields" which are effectively arbitrary packets of data associated by name. Although maintained by the framework, it does not possess or require any knowledge about the structure or contents of them. Finally, the framework also defines "state sets" which are named collections of fields, primarily used to modify the behaviour and state of individuals during the simulation. State sets can be used to represent the impact of policies on individuals, thus as policy changes, the associated state set is activated. There can only be one state set active at a time.

These five core abstractions have, thus far, proven a sufficient baseline for the construction of everything else.

In practice, the framework provides implementations of these constructs and other useful pieces of functionality. This includes data storage, serialisation, network communication, data input, and report output. In addition, there are a few code modules which, whilst not part of the framework itself (you can remove these modules and the framework will be unaffected), are still needed for various simulation components to function. These implement shared, but non-core, concepts such as abstract action representation and mortality.[2] The action system was added to provide a centralised mechanism for exposing operations on individuals, at the input data set level. Modules (see below) can expose trigger conditions to the simulation designer, allowing some predefined action to take place when said condition is met. For example, there could be a trigger in the event that an individual becomes infected by, or recovers from, a disease. This trigger might change the individual's current destination, or their active state set. On loading, modules register the actions which they implement with the action system, making them available to all other modules.

[2] Prior to the introduction of the mortality concept, peeps killed by the infection module would continue about their daily schedules. Whilst disturbing, this did not materially affect the results. Death is, it seems, widely useful but not always required.

### 3.2 The Simulacron Modules

A number of simulation modules have been implemented atop the framework. Each module implements a logically distinct process which can operate on cells, peeps, or both. The data associated with each module is maintained by the framework in one or more fields defined by that module. Presently, there are modules which implement infection spread, terrorist/police interactions, scheduling of actions, path-finding, and basic rule-based motivation inspired by BDI. There are also a number of smaller modules implementing simple functionality such as generation of censuses, peep state, etc. The underlying module design principle we have adopted is to focus on *abstraction*, rather than *specialisation*. This leads to a far greater capacity for re-use and adaptation to scenarios beyond those originally envisaged. This generality can involve greater computational cost, but this has not proven to be problematic in practice. We also believe that the added design and implementation work is overwhelmingly worth the effort.

**Scheduling** The scheduling module's purpose is to provide a way to define the movements and actions of peeps over time. The module provides two primary ways to control peeps: *cyclic schedules* and *event schedules*.

In both cases, schedules contain one or more schedule entries. Each of these entries consists of a trigger condition and a standard action. The trigger condition is always a time stamp, although the interpretation of its value differs between the two kinds of schedules. In a cyclic schedule, each day, of which there may be many, interprets time stamps as a relative "time of day,"[3] whilst entries in event schedules are interpreted as absolute times.

Although there is nothing in the design of the simulation framework that precludes interactive manipulation of data, such a facility has not yet been implemented. As a result, complex sequences of events are specified in the data set prior to beginning the simulation using the scheduling functionality. Changes in peep behaviour are stored as state sets, with schedules triggering transitions between these states. Changing the active state set can, in turn, cause a different schedule to be used for subsequent ticks.

**Infection** The infection module was designed to model a variant of the traditional SIR (Susceptible, Infected, Removed) infection model. The sequence of states supported by the model are: *Healthy Susceptible*, *Asymptomatic Latent*, *Asymptomatic Infectious*, *Heroic Infectious*, *Symptomatic Infectious*, followed by either *Immune* (which leads back to *Healthy Susceptible*) or *Dead*. "Heroic" is an original state which reflects the propensity for people to actively ignore or hide symptoms of an infection for various reasons. Consider the office worker who believes that "it's just a sniffle; nothing serious" or that they cannot afford to take leave from work.

In addition, the module features a conditional isolation feature which allows susceptible individuals to be sent to an isolation cell during specific period(s) of

[3] That is, they discard the "date" portion of the time stamp.

the day. An example is that sick children at a boarding school are not "detected" and isolated during the night, this will occur during the day when the teachers and other staff are awake. The module also has features for conditional infection, allowing the population to be divided into subgroups which can infect other, specific subgroups. This was designed to allow for basic multi-vector infections to be modelled.

Finally, support for multiple, concurrent infections has recently been added. Although it is not explicitly supported by the module, the presence or absence of one disease can influence an individual's susceptibility to another, through the state set mechanism.

**Transport** The transport module provides a generalised routing and path finding mechanism, allowing individuals to navigate through a sequence of goal destinations. The navigation functionality is provided by a simple A* algorithm. As the basic cell has no concept of size or location, these are added through this module to allow such path finding to take place. This has largely replaced the previous movement mechanism, which was instantaneous teleportation.

**Dispersion** The dispersion module was designed to provide a simple mechanism for non-deterministic movement of peeps. Over time, it has evolved into a more general mechanism for environmentally directed movement. The core functionality of this module is the random dispersion of peeps. Each cell with a dispersion field contains a set of one or more dispersion targets. On each tick, each peep in the cell is sent to a target randomly selected from this set.

The canonical example of this is a school playground. The playground is divided into four quadrants. Each quadrant is configured to disperse into the adjacent quadrants. With this arrangement, children peeps entering any part of the playground will randomly mill about until moved out of the playground by some other process.

In addition to the above, the dispersion module supports three additional mechanisms: masking, biasing and rate limiting. Masking allows the dispersal pattern to be varied depending on the specific peep being dispersed. For example, masking could be used to move wind peeps around an environment without causing person peeps to be affected. Biasing allows the relative probability of each target in the dispersion set to be adjusted. Rate limiting is used to control the amount of time between dispersal of successive peeps. This is often used to implement queues, barriers and other similar features.

**Induced Peep Actions (IPA)** The IPA module allows for arbitrary actions (as defined by the action system) to be performed on peeps. One use of this module is to implement detours by redirecting peeps entering a certain cell to a new destination. In conjunction with the state set mechanism, this allows directed evacuation behaviours to be initiated and simulated.

**Motivation** The motivation module provides a mechanism for peeps to interact with each other and their environment. The module allows peeps to have variables (called characteristics), which can be influenced by other peeps and the environment. Peeps may also have one or more if-then-else rules, which can execute actions. The values of a peep's characteristics determine which rules (and hence, actions) are executed. This has been used, for example, to model the spread of panic through a population. It does not provide for behavioural learning, being designed to be as simple as possible whilst still providing a useful level of fidelity. There is nothing to preclude the development of an additional module which supports learning and dynamic behaviour.

**Terrorist, Police, Citizen (TPC)** This module enables the modelling of simple terrorist interdiction scenarios. To that end, it divides the population into three broad classes. Terrorists attempt to remain undetected by the police until their appointed attack time, whereupon they undertake whatever action has been specified (defaulting to killing everyone in the vicinity). Police attempt to detect and detain terrorists. Citizens play no active part in the simulation as far as this module is concerned.

### 3.3 Associated Tools

**Data Set Template Processor (DSTP)** For the sake of flexibility, data set preparation is distinct from the simulation itself. Input data sets are plain-text XML files which completely define all entities and everything which is known about them; there are no emergent properties in the input. These files are the only input, aside from command-line parameters, which the simulation program accepts. The files are human-readable, allowing them to be written and modified by hand if necessary.

More commonly, however, they are constructed using templates. The template language is a set of XML pre-processing commands mixed with the raw data, interpreted by the DSTP program. These templates allow for the moderately straightforward creation of large populations. For example, instead of defining one thousand individuals, one can define a template that defines the statistical distribution of a population's properties, then sample 1,000 individuals from said population. The advantage of this method is that you do not lose the ability to manually construct unusual or special individuals. As with the generated data sets, the template files are plain text.

It is worth noting that the expansion of templates being independent of running the simulation also allows for random variation at two levels: a template can be re-instantiated with a different seed, producing a different, but similar, population. Alternatively, the same instantiated population can be re-simulated with a different simulation seed, leading to stochastic sampling of the behaviour spectrum of a fixed population.

Templates can also be extended with a simple built-in scripting language, or external programs. These allow for arbitrarily complex transformations to

be performed without the need for additional, explicit steps in the process. As an example, one scenario required the calculation of spatial properties for the environment, a task handled by a simple external program written in the Python programming language and executed by DSTP.

DSTP allows templates to range from unmodified, static data sets (data sets with patterns abstracted) through to complex, multi-file, modularised templates with domain-specific representations and command-line replaceable definitions.

**Refinery** This is used to estimate input parameters given known statistical measures. This is especially important for epidemiological purposes where suitable parameters, in the form we require, may not be available.

Refinery works by instantiating a population with parameters sampled from a pre-determined, broad distribution. Each instantiated population is then simulated a number of times. Simulated annealing is then used to refine the parameter ranges which are subjected to re-iteration. This process is repeated until convergence occurs, or until reasonable time or iteration constraints are exceeded.

Currently, Refinery can only be used to estimate a fixed set of parameters related to epidemiological modelling, but the principle is readily extensible to other parameters.

### 3.4 Simulacron in Practice

**Royal Naval School Influenza** [7, 13, 8] The first case we studied was the influenza outbreak at the Royal Naval School in Greenwich, England during 1920. This was chosen as there was very detailed information on the outbreak and the conditions in which it occurred. The school itself had roughly one thousand students attending it, almost all of whom lived on-campus and rarely left the school grounds, making it an ideal, real-world closed community.

The scenario employed three of the simulation modules: the scheduling module which handled each individual's daily routine, the dispersion module which was used to model playground behaviour, and the infection module which codified the spread of the infectious disease. This scenario produced close agreement with historical data, even prior to the introduction of Refinery.

**Australian Community Terrorism** [13] We have also applied Simulacron to the modelling of terrorist behaviours. There were two primary motivations for this work: to demonstrate the applicability of the modelling technique to problems outside of epidemiology, and determine whether the design of the simulator was sufficiently flexible to allow the existing framework and simulation modules to be adapted to a different purpose.

This involved the modelling of a statistically average Australian community which was subsequently used as the "background population" for the terrorism scenarios. The properties of both the individuals and the family units were chosen to match statistical data from the Australian Bureau of Statistics' 2006 census for New South Wales. Individuals moved about the community according to

individually constructed schedules which emulated six groups of people: full-time employed, part-time employed, unemployed/home workers, primary school students, secondary school students, and infants. The distributions of individuals in family groups, residences and workplaces were also statistically matched.

Two scenarios were constructed atop this background community. The first involved the release of an infectious biological agent at a specific time and place within the community, with the spread of this agent measured. The infection module was used for this scenario, the biological weapon modelled as a stationary individual who was scheduled to become extremely infectious at the given attack time for a short duration. Our results suggest that location of release is a significant factor in determining the number of subsequent infections. This is hardly surprising given that different locations will have greatly varying traffic levels and patterns. Such simulations could be of interest for the purposes of risk assessment, provisioning resources to cope with an attack, and the effectiveness of potential interventions.

The second scenario examined the effectiveness of a simple interdiction strategy against terrorist activity. The goal of the terrorist in this scenario was to avoid detection until arriving at a club where they would detonate a bomb, killing all individuals in the building. Prior to the attack, the terrorist would engage in "normal" activities, consistent with others in the community. This was intended to mimic an individual attempting to avoid drawing attention to themselves. Police officers would move about the community on pre-planned patrol routes. The TPC module was initially developed for this scenario.

**Amoy Gardens SARS** [1, 10] This simulation modelled the spread of severe acute respiratory syndrome (SARS) through Amoy Gardens in Hong Kong during March and April of 2003. This was chosen because it allowed us to study the effect that enforced isolation has on the spread of an infection, and it served as an opportunity to test and demonstrate our ability to model multi-vector infections. To that end, a synthetic community representing the residents of Block E was constructed, using demographic and work characteristics for a typical Hong Kong community. Although work on the scenario has not yet been completed, preliminary results demonstrate that the aerosol dispersal of infected droplets does play a significant role in such outbreaks.

It is worthy of note that both the humans and aerosol particles are modelled using peeps. The significantly different roles and behaviours of these two are managed via the masking mechanisms in the infection and dispersion modules.

These simulations were run on a relatively low-spec consumer laptop with one hardware thread. They involved approximately 15,000 individuals, who were simulated with the infection, scheduling and dispersion modules.

**Sydney Central Station Terrorism** [9] Our most recent completed work was in modelling a hypothetical bomb blast at Central Station in Sydney, Australia. The purpose of this scenario was to perform a more thorough and involved test of the motivation module, provide an environment for an initial implementation of

a more realistic path-finding movement module, and to demonstrate and assess our tools' capacity for modelling a realistic transportation hub.

This scenario involved the modelling of the upper level of Central Station. A synthetic commuter population of approximately half of peak-hour traffic was also created and simulated. Normal commuter behaviour was simulated for one hour before the introduction of an assailant peep. The assailant moved through the station to a pre-determined location before detonating an explosive device. The explosion killed or injured everyone in the immediate vicinity and was normatively equivalent to a low-yield backpack IED. The simulation continued until the two hour mark in order to observe how the commuters responded to this event.

The motivation module allowed the modelling of post-blast commuter behaviours consistent with those observed in similar, real-life events.

The explosion and subsequent effects of the bomb were modelled with the infection module by representing it as a high-lethality infection with no latent, asymptomatic or symptomatic periods, and no cross-infection.

Although this scenario was a proof-of-concept, it demonstrates a realistic capacity for our approach and tools to model events of this nature. With a modular framework and generalised components, a reasonable facsimile of a real-world location was simulated with a meaningful number of individuals, all on a single desktop computer.

These simulations were run utilising a single core of a stock Intel® Core™ i7 860 CPU, taking an average of 50 minutes to simulate 2 hours with a temporal granularity of 5 seconds. These scenarios involved 10,000 commuters, simulated with a combination of infection, scheduling, dispersion, induced peep actions, path finding and motivation. As a point of comparison, a similar agent-based simulation involving evacuation of individuals from the International terminal of the Los Angeles International Airport involved just 200 representative individuals [19]. Although we do not have information regarding the computational resources required by this, the limited number of individuals modelled strongly suggests that they were not inconsiderable.

## 4 Conclusion

We believe that the success of Simulacron across multiple simulation domains demonstrates the value of the strategy we have adopted. It provides a light-weight mechanism for the implementation of heavy-weight simulations, through a focus on generality and modularisation, and enables reusability and composability.

The goals of the research team drive development from two directions. One is the provision of increasing flexibility and scope of modules, the other is the creation of more complex simulations. The ability to address both of these within a single team reflects the advantages and the challenges of working in a multi-disciplinary environment. The challenge of the project is to pursue the first without needlessly complicating the construction of simulations, and the second

without creating a swathe of over-specific modules. To date, we believe these seemingly contradictory goals have been achieved, and we see no strong evidence that this will not continue to be the case.

In the near future, we plan to conduct an extensive re-write of the Simulacron framework, intended to further improve its performance and to make it more accessible to other researchers. This development will allow larger, more complex simulations across a broader range of problem domains. This will also facilitate further collaboration with outside agencies, increasing the breadth of utility of the framework. Ideally, this would include such agencies actively developing modules atop the framework.

Our design methodology thus far has been an organic one, meaning that we do not have any fixed direction for the future development or expansion of our simulation approach. As most of the enhancements made to-date (such as the introduction of state sets) were prompted by needs arising from the development of the software, it is hard to predict the details of future changes. However, the guiding principle of "keep it simple; if you can't, keep it general" is unlikely to change.

## References


1. Outbreak of Severe Acute Respiratory Syndrome (SARS) at Amoy Gardens, Kowloon Bay, Hong Kong; Main Findings of the Investigation. Tech. rep., Health Department of the Hong Kong Special Administrative Region of the People's Republic of China Government Information Centre (April 2003)
2. Balci, O., Arthur, J.D., Ormsby, W.F.: Achieving reusability and composability with a simulation conceptual model. Journal of Simulation 5(3), 157–165 (2011), http://www.palgrave-journals.com/doifinder/10.1057/jos.2011.7
3. Birkin, M., Wu, B.: A Review of Microsimulation and Hybrid Agent-Based Approaches. In: Heppenstall, A.J., Crooks, A.T., See, L.M., Batty, M. (eds.) Agent-Based Models of Geographical Systems, pp. 51–68. Springer Netherlands (2012), http://dx.doi.org/10.1007/978-90-481-8927-4_3
4. Brouwers, L., Boman, M., Camitz, M., Mäkilä, K., Tegnell, a.: Micro-simulation of a smallpox outbreak using official register data. Euro surveillance 15(35), 1–8 (Sep 2010), http://www.ncbi.nlm.nih.gov/pubmed/20822732
5. Chen, X.: Microsimulation of Hurricane Evacuation Strategies of Galveston Island. The Professional Geographer 60(2), 160–173 (Jan 2008), http://www.tandfonline.com/doi/abs/10.1080/00330120701873645
6. Crooks, A.T., Heppenstall, A.J.: Introduction to agent-based modelling. In: Heppenstall, A.J., Crooks, A.T., See, L.M., Batty, M. (eds.) Agent-Based Models of Geographical Systems, pp. 85–105. Springer Netherlands (2012), http://dx.doi.org/10.1007/978-90-481-8927-4_5
7. Dudley, S.F.: The Spread of "Droplet infection" in Semi-Isolated Communities. H. M. Stationery Office (1926)
8. Green, A.R., Zhang, Y., Piper, I.C., Keep, D.R.: Application of Microsimulation to Disease Transmission and Control. In: Proceedings of the 2011 MODSIM International Congress on Modelling and Simulation. pp. 919–925 (2011), http://www.mssanz.org.au/modsim2011/B2/green.pdf

9. Keep, D., Piper, I., Green, A., Bunder, R.: Modelling of Behaviours in Response to Terrorist Activity. In: Proceedings of the 2011 MODSIM International Congress on Modelling and Simulation. pp. 475–481 (2011), http://www.mssanz.org.au/modsim2011/A6/keep.pdf
10. Keep, D.R.: Generalised Microsimulation. Ph.D. thesis, University of Wollongong (2012)
11. Orcutt, G.H.: A New Type of Socio-Economic System. The Review of Economics and Statistics 39(2), 116 (May 1957), http://www.jstor.org/stable/1928528?origin=crossref
12. Perez, P.: Private communication (2012)
13. Piper, I., Keep, D., Green, T., Zhang, I.: Application of Microsimulation to the Modelling of Epidemics and Terrorist Attacks. Informatica 34, 141–150 (2010), http://www.freepatentsonline.com/article/Informatica/232715912.html
14. Sauerbier, T.: UMDBS - A New Tool for Dynamic Microsimulation. Journal of Artificial Societies and Social Simulation 5(2) (2002), http://jasss.soc.surrey.ac.uk/5/2/5.html
15. Smith, L., Beckman, R., Baggerly, K., Anson, D., Williams, M.: TRANSIMS: Transportation analysis and simulation system. Tech. Rep. 26295, Los Alamos National Lab., NM (United States) (1995), http://www.osti.gov/energycitations/product.biblio.jsp?osti_id=88648
16. Spielauer, M.: What is Social Science Microsimulation? Social Science Computer Review 29(1), 9–20 (May 2010), http://ssc.sagepub.com/cgi/doi/10.1177/0894439310370085
17. Statistics Canada: Modgen (2011), http://www.statcan.gc.ca/microsimulation/modgen/modgen-eng.htm
18. Stroud, P., Del Valle, S., Sydoriak, S., Riese, J., Mniszewski, S.: Spatial Dynamics of Pandemic Influenza in a Massive Artificial Society. Journal of Artificial Societies and Social Simulation 10(4), 9 (2007), http://jasss.soc.surrey.ac.uk/10/4/9.html
19. Tsai, J., Fridman, N., Bowring, E., Brown, M.: ESCAPES-Evacuation Simulation with Children, Authorities, Parents, Emotions, and Social comparison. In: Tumer, Yolum, Sonenberg, Stone (eds.) Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Systems Innovative Applications Track (AAMAS 2011). pp. 457–464 (2011), http://www.aamas-conference.org/Proceedings/aamas2011/papers/D3_G57.pdf
20. Wolshon, B., Resiliency, T., Lefate, J., Naghawi, H., Montz, T., Dixit, V.: Application of TRANSIMS for the Multimodal Microscale Simulation of the New Orleans Emergency Evacuation Plan. Tech. rep., Louisiana State University (2009), http://www.ntis.gov/search/product.aspx?ABBR=PB2011107618
21. Wu, B.M., Birkin, M.H., Rees, P.H.: A Dynamic MSM With Agent Elements for Spatial Demographic Forecasting. Social Science Computer Review 29(1), 145–160 (May 2010), http://ssc.sagepub.com/cgi/doi/10.1177/0894439310370113