

MIXED-MODE PANDEMIC MODELING

Peter Bosch

Highpoint Software Systems
S33 W35501 Meadow Trail
Dousman, WI 53118, USA

ABSTRACT

This paper describes a mixed mode approach to simulating a global pandemic. It describes the architectural and design decisions as well as the data sources, development tools and processes in each of the three simulation domains - System Dynamics (SD), Discrete Event Simulation (DES) and Agent-Based Modeling (ABM). The encompassing model uses a network of SD models, each representing a 30 km² grid cell of Earth's surface, with attributes describing population density as well as economic, political, sanitary and healthcare capabilities. Governments and outbreak response teams are represented by software agents, making decisions about resource allocations and performing vaccination drives. A discrete event engine drives the SD models and the agents, as well as any prescribed response actions. Where possible – and it was almost always possible – Open Source data and software is used.

1 INTRODUCTION

Mixed-mode simulation seems to be difficult to attain with sufficient robustness in flexibility and feature set. Some vendors of so-called “Workbench” tools (Bosch 2003b) support it, but as soon as things get complex, the developer has to write code. (Brailsford et al. 2018). We believe that acknowledging that common, and often unavoidable, complexity, and starting with software first principles and an architecture tailored to the application domain, instead of later being predictably forced into coding within the structure and lack of internal visibility imposed by a general purpose tool vendor to whom one has already committed, is often the most productive approach – and provides the best chance at success in a broad range of more complex scenarios.

This topic is addressed in (Bosch 2003b), where we describe the reasoning behind the assertion that for certain types of simulations, an open architecture approach holds benefits that should be considered. We also discuss scenarios wherein a workbench-style tool holds the advantage in quick, easy achievement of usable results.

2 PROBLEM DOMAIN

Epidemics occur and proceed according to the dictates of three sets of characteristics, the disease pathology, the geographical structure into which the disease is introduced, and the socioeconomic attributes of the locales in which it occurs. (Salama 2005; Healthline 2016) We want to create a tool that allows exploration of global epidemics and the factors that affect them, simultaneously creating an example of a mixed-mode simulation. In this section, we describe the real-world characteristics being modeled.

2.1 Disease Characteristics

A disease is characterized in a number of ways that relate to its performance in an epidemic. Diseases exist in two states – endemic and epidemic. Endemic diseases happen at a more or less steady, expected rate. When that rate rises unexpectedly, it is said to be in an epidemic state. (Dicker et. al 2011) Eradication is

an effort that aims to achieve total elimination of the disease. This model does not address eradication, though it is a future aim.

Transmission describes the initiation of disease in a new host as a result of its presence in a preceding host. Transmission can be direct or indirect. Direct transmission requires physical contact with skin, mucous membranes or bodily fluids, or in some cases, inhalation or ingestion of a droplet from a sneeze or other exhalation of the infected host. Many diseases rely on a chain of inter-species but still direct transmissions, such as from human to insect or other animal to human. Indirect transmission occurs when the pathogen exists outside a living host for an extended time – such as in water or on a surface such as a doorknob, or suspended in air before entering a new prospective host. (Healthline 2016) Many pathogens only survive outside the host for a limited time. Each of these mechanisms carries a different likelihood of successful transmission.

Diseases typically repeat a consistent cycle from one host to the next. The cycle is described through several attributes. The serial interval is the length of time between the onset of one case and the onset of a case caused by infection from that first case. The incubation period is the length of time between the invasion by the pathogen and the appearance of disease symptoms. The latent period is the amount of time between the invasion by the pathogen and the onset of infectiousness. Infectiousness may occur before or after the appearance of symptoms. The infectious period is the amount of time that the disease is communicable.

Some diseases rely on genetic or other characteristics such that certain individuals are not susceptible to the disease. Additionally, some diseases, once a person has contracted them and recovered, have a reduced or nonexistent ability to impart reinfection.

Epidemiologists use the ‘Transmission Probability Ratio’ (TPR) to represent the likelihood that an uninfected individual, after having come in contact with an infected individual, will contract the disease. There are four such probabilities, p_{00} , p_{01} , p_{10} , and p_{11} , accounting for the probabilities each of the infected and the uninfected having undergone vaccination or not. If there is no such vaccine, then the only meaningful TPR is p_{00} , pertaining to unvaccinated source and target hosts.

2.2 Geographical Structure

At the outset, this model was intended to model global pandemics, so global geography is incorporated into the model, extracted from the dataset at (CIESIN 2018) which accounts for every 30 km² cell on the surface of the planet, and provides its population and land/water status. This data set served a dual purpose – both the surface geography of the planet, and the population thereupon. From (Kelso 2018) we obtain cultural (i.e. national) boundaries.

2.3 Socioeconomic Characteristics

We incorporate many socioeconomic characteristics in the model, mostly derived from data contained in (CIA 2019). As pointed out in section 5.2, the assertions in this section are hypotheses, intended to speed the development of the model. We do this because the purpose of this paper is not so much a rigorous proof of a theorem as it is expository of a modeling technique.

A disease spreads among people who interact. At a local level, they are a community, at an organizational level, a society. Communities, as an approximation, are called ‘cells’ in this model and by virtue of the data set employed, are defined as a 30km x 30km section of the planet’s surface. They are geographically local, and as a large population may contain many communities that diffuse into one another, adjacent communities have a blending function – migration and oscillation of people between them. (Noulas et al. 2017)

We believe that a society or community’s quality of healthcare dictates the degree to which existing vaccinations are preventively applied and the breadth and depth to which that application permeates the population. Furthermore, as an epidemic begins to emerge (i.e. the actual rate of incidence rises from the expected rate,) the quality of a healthcare system is one key element in the detection of that epidemic. And

when an epidemic has taken hold, that quality of healthcare results in better capabilities toward distributing and administering any vaccines that are available.

We believe that societal economic standing can result in better ability to discover, manufacture, stockpile, and distribute vaccinations and teams to administer them, without having to rely on foreign aid and intervention. That ability gives a population more control over its response to an epidemic. It can export or retain disease-fighting capability, or choose a combination of the two.

We believe that better societal sanitation – in the forms of cleaner personal habits, availability of clean water and better handling of wastewater – results in reduced transmission of some diseases.

We believe that more stable societies engender more efficient and safer governmental health and sanitation efforts of both preventive and mitigative natures.

Populations migrate and oscillate (i.e. commute). Migration carries small portions of the population long distances, and oscillation suggests a much smaller radius of motion that is applied essentially every day. Thus, a small number of infectious persons may travel a long distance each day, and a large number will travel a small distance. (Balcan et al. 2009) Commuting is somewhat diminished by national borders and geographic boundaries such as rivers and mountain ranges, though neither of these effects is modeled yet in this project.

3 DESIGN APPROACH

3.1 Structural Design

We split Earth's surface up into approximately 30 km x 30 km grid squares to model population. In each grid square we represent the progress of a disease using a rather simple System Dynamics model as shown in Figure 2. At startup, initialization code at each country level examines its socioeconomic attributes described in 2.3, and determines levels of existing vaccination, stockpiles, and foreign aid – both committed and dynamic. Committed foreign aid is human services foreign aid that is committed to particular countries, and that serves to raise the quality of healthcare in the target country. A policy agent is created for each country that represents its national government, and decides when to implement various policies having to do with quarantine, travel bans, requests for foreign aid & both native and foreign epidemic combat teams, granting of other countries' requests, distribution of vaccine stockpiles within and outside its borders, and commandeering of industrial capability to improve its ability to manufacture more vaccines. Each day, the model imparts the effects of oscillation and air travel, moving a percentage of the population a short distance into neighboring grid squares, and a much smaller proportion between airports and their surrounding grid squares.

3.2 Data Sources

3.2.1 Spatial

At the lowest level, we create a geographical model using country boundaries from (Kelso 2018). We then create a population model using population density for each (approximately 30km x 30 km) grid square, obtained from the 15-minute of arc (approximately 30 km x 30km) dataset at (CIESIN 2018). Each grid square is assigned a nominal country owner, based on its latitude and longitude. We then create a socio-politico-economic model. From (CIA 2019), we obtained a mapping of countries to a number of characteristics such as physicians per 1000 persons, life expectancy, infant mortality rate and health expenditure per capita (which we roughly translate into a quality of healthcare rating,) Per Capita Gross Domestic Product, which we translate – again, roughly – into an ability to research, manufacture, stockpile and distribute vaccines, as well as the level of preparation of a country for diseases, and availability of dynamic foreign aid. Drinking water source and sanitation facility access impact the transmissibility of the disease. From (World Bank 2017), we set initial levels of committed foreign aid. We map literacy ratings and percentage of population with electricity, arguably, to societal stability, and from there, also arguably,

to effectiveness of disease-fighting efforts. Finally, in creation of a transportation model, we obtained airport locations from (Patokallio 2018), airline flight parameters such as origin, destination, and equipment, which, paired with equipment seating configurations from (FAA 2018) and an assumed average passenger loading of 80%, developed an estimate for global commercial air passenger movement.

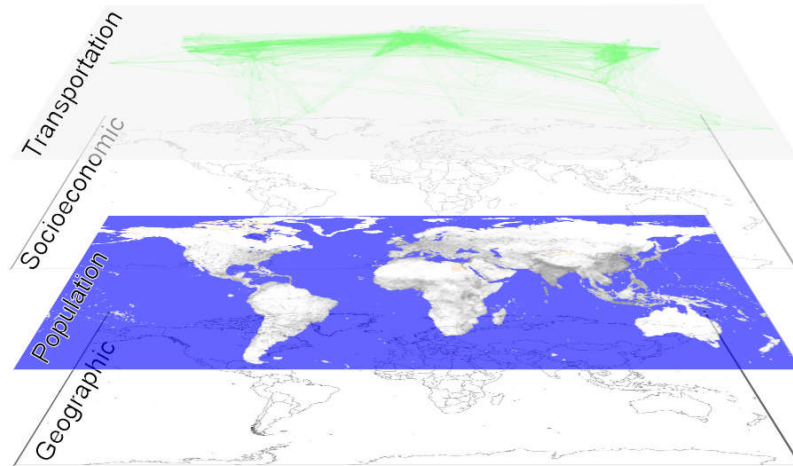


Figure 1: Spatial data source models.

3.2.2 Disease

We use a variation of the well-known SIR disease model (Smith et. al 2004), shown below in Figure 2. It has been incorporated into a human lifecycle model and enhanced to accommodate a statistically-possible (i.e. genetic) natural immunity. In addition, it is expanded to show a contagious asymptomatic, contagious symptomatic, and non-contagious but still infected stage. Natural death and death by disease are represented as well, in order to gather key data. This disease cycle is defined in a pair of C# files, one generated from XMILE as described previously, and one with hand-coded behaviors. In most cases, only the generated file needs to be replaced, in order to model a different disease.

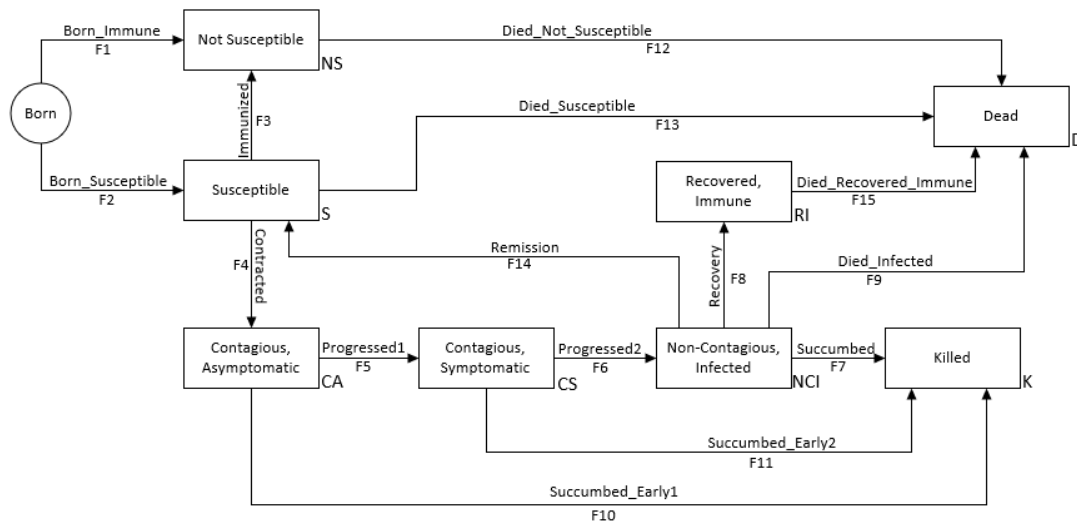
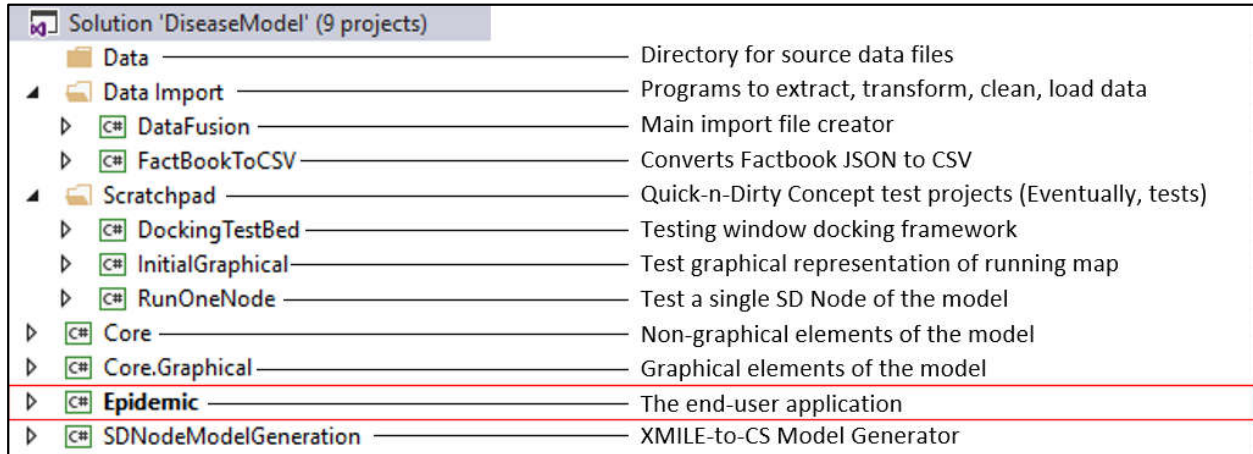


Figure 2: SIR disease model.

4 TECHNICAL IMPLEMENTATION

We chose support libraries on the basis of their free (as in “free speech”) or if unavailable as such, their free (as in “free beer”) standing (FSF 2019). All code, data and support libraries for this work are available for download from the internet as of the time of publication of this paper, and many are automatically retrieved using the NuGet library management system (Nandwani et. al 2018). The subject of this paper is contained in a Visual Studio (Community-Edition-Compatible) solution. In that solution are a number of projects, shown – with descriptions – in Figure 3. External projects and libraries are specified as NuGet packages, and will be automatically imported into the solution.



Solution 'DiseaseModel' (9 projects)	
Data	Directory for source data files
Data Import	Programs to extract, transform, clean, load data
DataFusion	Main import file creator
FactBookToCSV	Converts Factbook JSON to CSV
Scratchpad	Quick-n-Dirty Concept test projects (Eventually, tests)
DockingTestBed	Testing window docking framework
InitialGraphical	Test graphical representation of running map
RunOneNode	Test a single SD Node of the model
Core	Non-graphical elements of the model
Core.Graphical	Graphical elements of the model
Epidemic	The end-user application
SDNodeModelGeneration	XMILE-to-CS Model Generator

Figure 3: Project structure.

4.1 External Libraries

4.1.1 Sage

Sage is the open source simulation software library we chose – primarily on the basis of familiarity, though it is robust in its feature set – to implement this application. It is an open source simulation library, written in C#, with support for Discrete Event and Agent-Based Simulation, and emerging support for System Dynamics. Mechanisms for each will be described in this paper. Sage was originally called HighMAST. (Bosch 2003a). Primarily, in this case, Sage is used for its event calendar, or “Executive”, and its SD underpinnings and tools. It is automatically installed into the solution via a NuGet reference. Available at <https://github.com/SageSimulations/Sage/>.

4.1.2 DockPanelSuite

DockPanelSuite is an open source window management library that allows the docking, undocking and automatic slide-away hiding of document and tool windows. It supports multiple themed appearance “skins,” and is automatically installed into the solution via a NuGet reference. It is available at <http://dockpanelsuite.com/>.

4.1.3 OxyPlot

OxyPlot is an open source, cross-platform plotting library for .NET. It is automatically installed into the solution via a NuGet reference. It is available at <https://www.oxyplot.org/>.

4.1.4 Other Included Libraries

A few more libraries are used with a lesser footprint, but should be mentioned as an indication of the broad availability of libraries to serve secondary functions.

- CSVHelper reads CSV files and presents a tabular representation to code. It is used in data import functions in the DataFusion and Epidemic projects, and data export in the FactBookToCSV data import project. It is automatically installed into the solution via a NuGet reference. Available at <https://joshclose.github.io/CsvHelper/>.
- Newtonsoft.Json reads JSON (JavaScript Object Notation) files and presents a tree-based object hierarchy to code. It is used in the FactBookToCSV data import project, and is automatically installed into the solution via a NuGet reference. Available at <https://www.newtonsoft.com/json>.
- Wibci.CountryReverseGeocode is a library that, given latitude and longitude, gets the country or American state name where the point is located. It is automatically installed into the solution via a NuGet reference. Available at <https://www.nuget.org/packages/Wibci.CountryReverseGeocode/>.
- ANTLR is a set of libraries that can be used to create a translator. Such a translator is a part of the SD Model generation subsystem of Sage. Available at <https://www.antlr.org/>.

4.2 Modeling Domains

The Epidemic application incorporates Discrete Event Simulation in the cyclical invocation of disease model state and airline passenger travel updates. Each grid square contains a System Dynamics model, coordinated and networked to emulate the growth or decline of a global population's contracting a disease, and Agent-Based Modeling is represented in software agents that emulate governments and outbreak response teams in their reactions to the epidemic, and influence the attributes of grid squares' SD models.

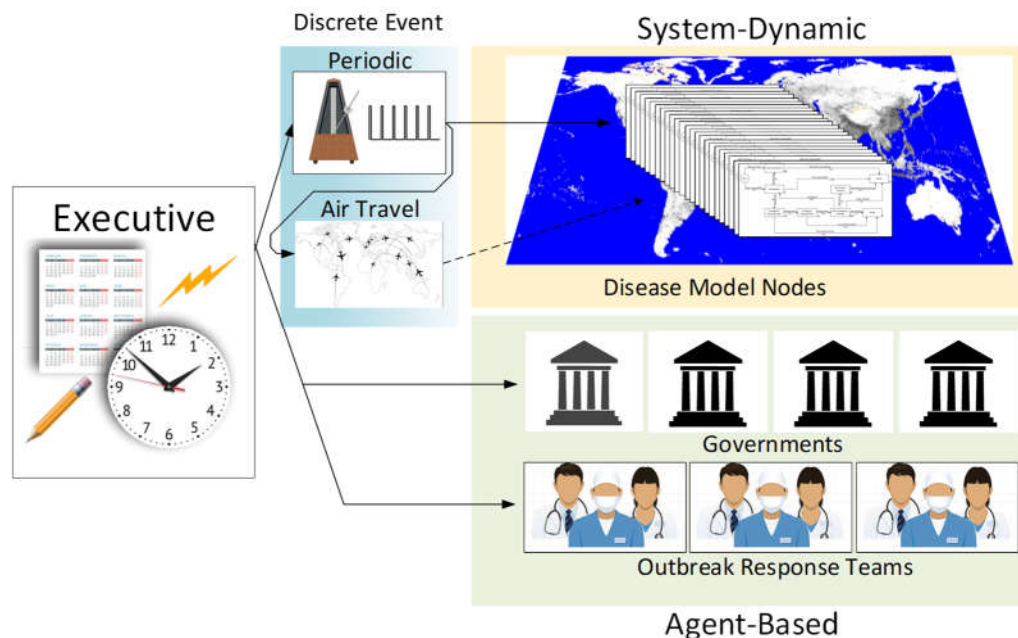


Figure 4 : Simulation modes.

4.2.1 System Dynamics

System Dynamics is an approach often used to model the progress of a disease moving through a population. The model we use is a relatively simple representation of a disease path, and is shown in Figure 2, though more complex representations are easy to implement. Model structure, as well as stock initialization, flow value and auxiliary expressions are specified as source code in an XMILE specification.

The model is generated from an XMILE specification, using a tool that is a part of the Sage library toolset. The mechanism for doing so is described in Figure 5. The example shown uses the classic Lynx-Hare Model (Volterra, 1928) for System Dynamics – provided as a sample model with the SD tools, Ventity, Simile and InsightMaker. An XMILE representation is used to generate a software representation of the Lynx-Hare model. All expressions in that model are processed and translated via an ANTLR grammar translator, to C# code. The software model is then the subject of a T4 engine's (Warren et. al 2016) generation of a textual representation of a C# class file. In our Epidemic Modeler, we automated this code generation in the SDNodeModelGeneration project (see Figure 3) and include the resultant DiseaseNode code by reference as a partial class in the Core project. The other half of the DiseaseNode class accommodates the more free-form elements like the constructor and initialization. There is one instance of the DiseaseNode model per cell, and data in the DiseaseNode model draws heavily from instances of Disease, Locale and Policy classes which are assigned and modified for each cell.

Note that DiseaseNode states are programmatically immutable. This mechanism can be used in a Functional Programming fashion, as shown in the source snippet in Figure 2, in which all states are retained, or it can be incorporated into, and driven by, a DES model as we have done in this project.

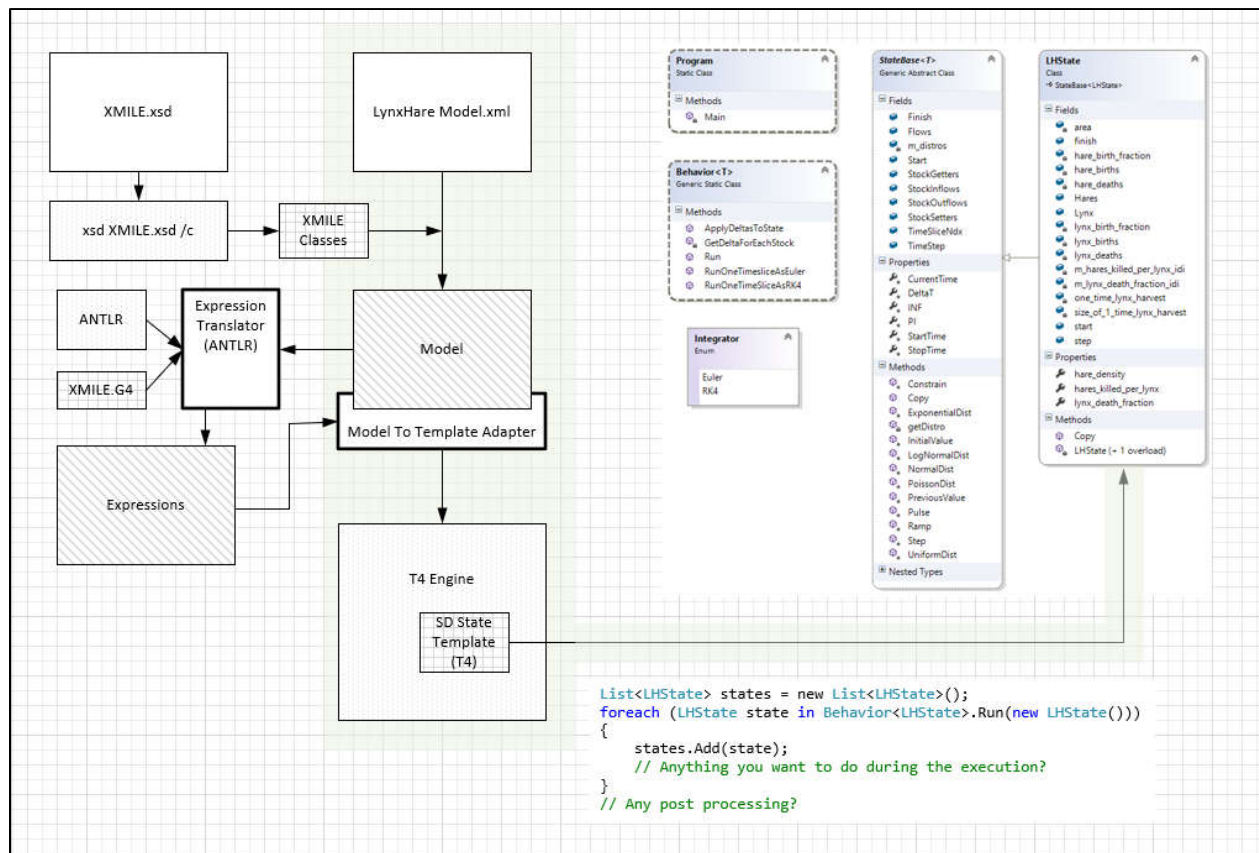


Figure 5: SD model class generation.

4.2.2 Agent Based Modeling

This project contains two examples of software agents. First, there is a “NationalGovernment” object for each country. With each cycle, i.e. daily, it surveys the disease state of the cells that fall within its boundaries, consults a list of countries and their participation levels in the outbreak, and configures a domestic vaccination effort according to its GDP and a perceived level of urgency. At the outset, vaccination worldwide is at a base level, multiplied by the country’s healthcare effectiveness rating. If the NationalGovernment has OutbreakResponseTeam (ORT) agents – representing such entities as those managed by the US Centers for Disease Control, it may dispatch them to outbreak areas, where they augment existing epidemic fighting facilities. All countries in the economic G7 have such teams, in proportion to their Gross Domestic Product. An ORT, once deployed to a country, makes its own decisions as to where in that country to work, with travel times taken into consideration. Since travel times within a country are almost always in fractions of a day, and the default time period for the SD model is one day, non-fixed time periods are necessarily used by ORT agents. At present, this logic, and the concept itself, are rather notional, but the implementation is fully functional.

4.2.3 Discrete Event

The entire simulation is driven by a discrete event engine. Each new SD timeslice computation is initiated by an object within the engine called a “metronome,” which delivers uniformly periodic events. Following each such update, population redistribution based on air travel data is implemented. Finally, governments and response teams are notified, and each is able to register for intraperiod notifications – for example, if a response team feels that only another 12 hours is needed in one location, it can commence its travel to the next location halfway through a (fixed) timeslice for the disease model.

5 RESULTS

5.1 Runtime

Below, see a series of screen captures from the running application. The simulation begins on January 1st, 2019, and in Figure 6, on January 6th, we have just introduced the disease into the southeastern United States. Air travel is as-yet unaffected.

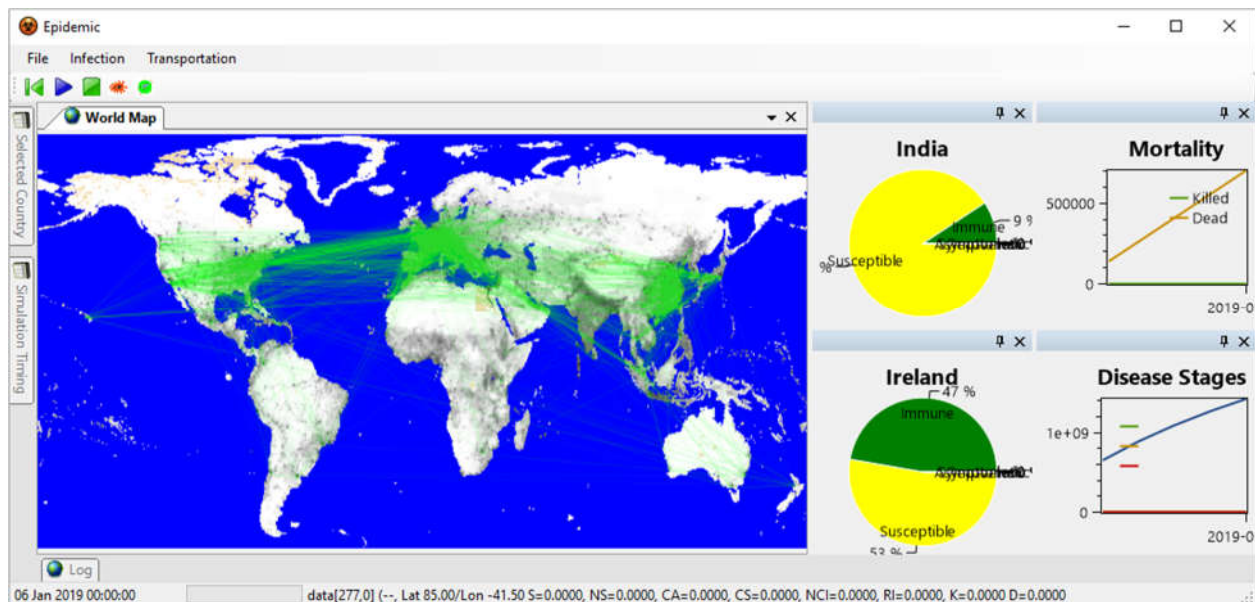


Figure 6: A healthy world - immunizations proceeding.

In Figure 7, on January 10th, the disease has made its way throughout the southeastern United States, and infected passengers are traveling by air to other cities. Air travel is still unrestricted.

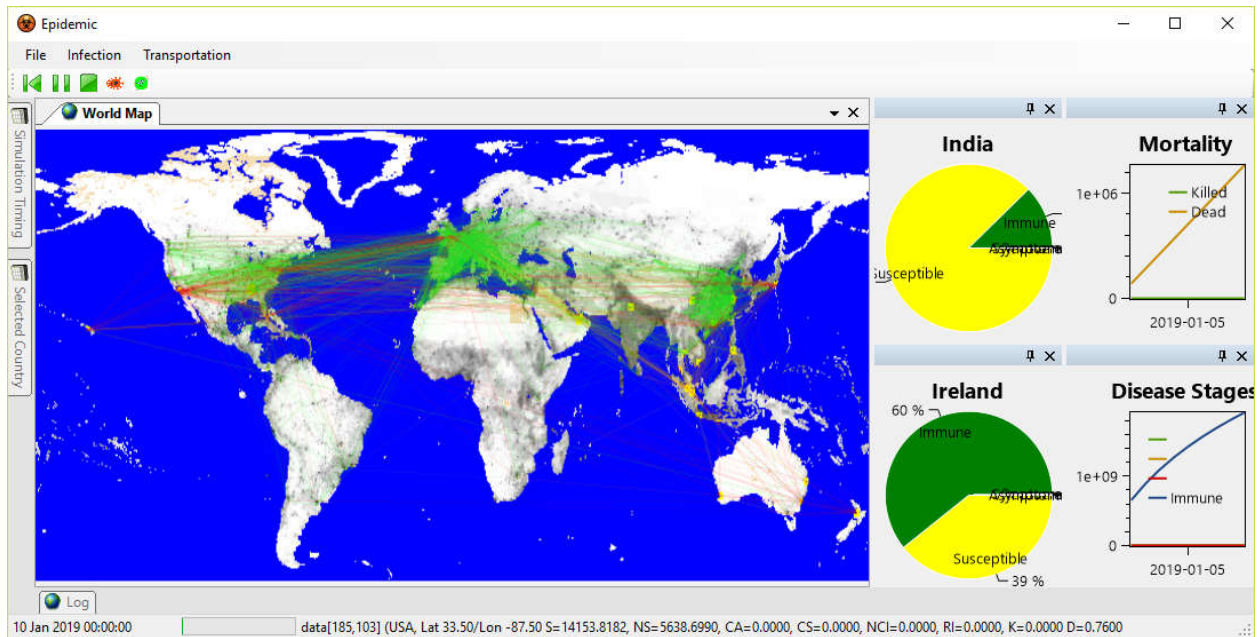


Figure 7: Air travel propagating the disease.

In Figure 8, on January 17th, over 200,000 have been killed, and a restricted air travel policy is in place. Running the model, one would see text in the log window that shows that over the course of several days, all air travel has been shut down.

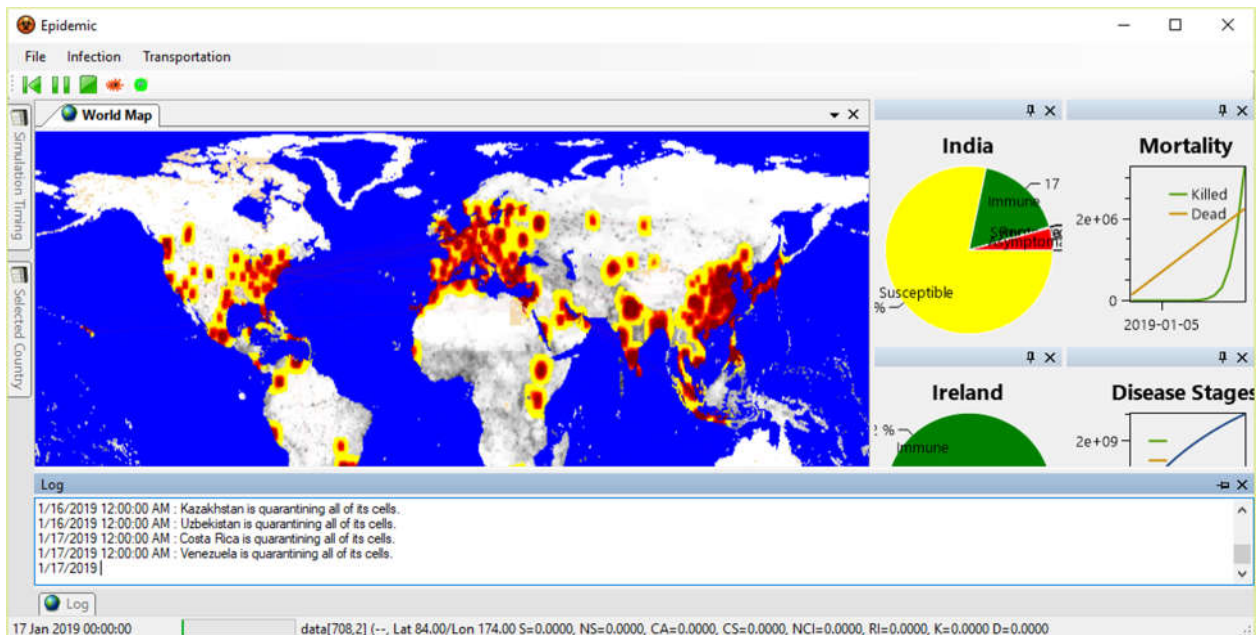


Figure 8: Restrictive air travel policy in place.

Finally, in Figure 9, it is May 9th. A late stage epidemic is shown, with substantial global mortality of over 2 billion, including almost 50% of the population of India.

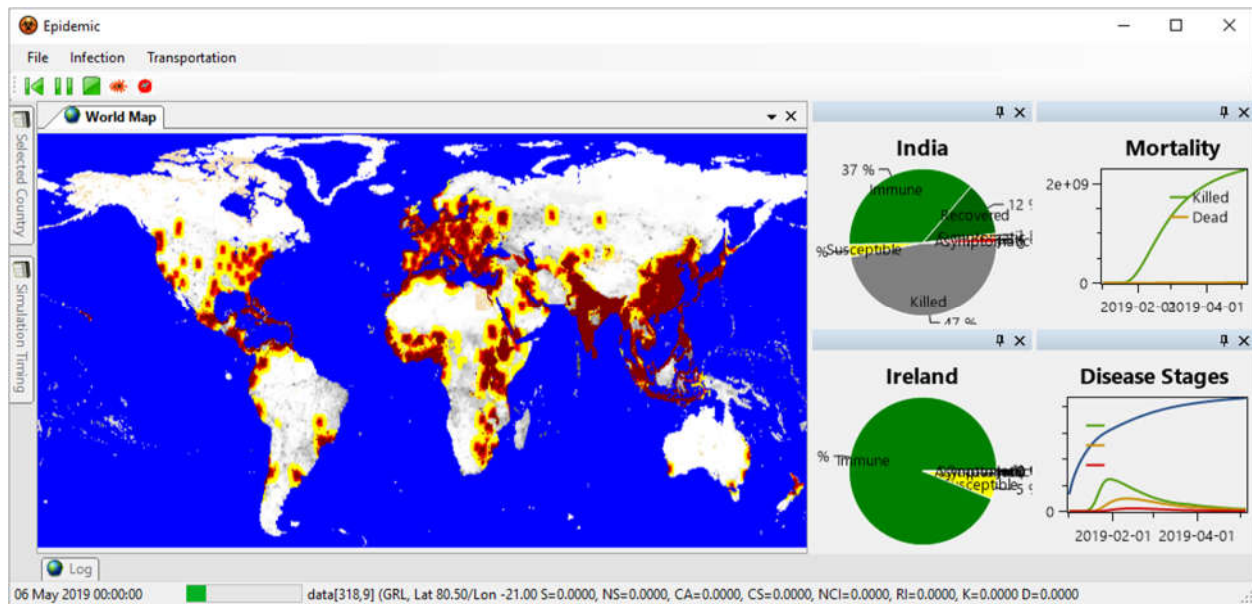


Figure 9: Late stage epidemic.

5.2 Verification & Validation

At present, this is a proof-of-concept effort, with verification and validation yet to occur. We have made numerous statistical and structural assumptions, many as placeholders for deeper analytical work in the future and some simply to support initial functionality (especially those in section 2.3). As such, we understand that the results of executing this simulation are not useful for real-world prediction, but that is not (yet) the purpose of this project. The next stage, if executed, will be to analyze, and if necessary, adjust, the assumptions and equations that translate from hard facts to driving variables.

6 ASSUMPTIONS

At present, healthcare is assessed per country rather than per region, so for example, vaccinations in rural and suburban areas of a given country are the same. We model only the ~27,000 most populous air routes, so some parts of the planet get no traffic, such as Madagascar. We model air travel, but not sea, rail or road traffic (road traffic is modeled implicitly via a cellular diffusion mechanism.)

Outbreak response teams are autonomous, but perhaps should come in groups, each under the control of an overarching authority such as the World Health Organization or Médecins Sans Frontières that at least attempts to coordinate with other organizations and governments.

We ignore travel explicitly from airports outward into the countryside, choosing to represent this in the diffusion mechanism that moves the population at large.

7 CONCLUSION

We advocate simulation development in the same manner that much other software is developed – from an architecture-first perspective, using appropriate componentry, rather than as a tailoring of an existing package. While either approach can be – and often is – employed in software development, the componentry-based solution, as with this solution, can enable quicker, better-tailored solutions than workbench-type applications. In this case, the selection of a simulation component into a free-form

architecture allowed a smooth integration and employment of Discrete Event, Agent Based, and System Dynamic methodology in the same application. Much has yet to be done with this work, including easing configuration, analysis and choices of policies. In addition, each of the assumptions mentioned in section 6, above, engenders some potential improvement. The source code for this project can be obtained at <https://github.com/SageSimulations/Epidemic>.

REFERENCES

- Balcan, D., V. Colizza, B. Gonçalves, H. Hu, J. J. Ramasco, and A. Vespignani. 2009. "Multiscale Mobility Networks and the Spatial Spreading of Infectious Diseases". *Proceedings of the National Academy of Sciences of the United States of America* 106(51):21484–21489.
- Bosch, P. 2003a. "Introduction to HighMAST". In *Proceedings of the 2003 Winter Simulation Conference*, edited by S. Chick, P. J. Sánchez, D. Ferrin, and D. J. Morrice, 1852–1859. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers.
- Bosch, P. 2003b. "HighMAST Versus 'Workbench' Style Simulation Tools". Waukesha, WI: Highpoint Software Systems, LLC. <http://www.highpointsoftware.com/documents/comparisonstoworkbench.pdf>, accessed 23rd June 2019.
- Brailsford, S., T. Eldabi, M. Kunc, N. Mustafee, A. Osorio. 2018. "Hybrid Simulation Modelling in Operational Research: A State-of-the-art Review". *European Journal of Operational Research* 278(3):721–737.
- CIA. 2019. Central Intelligence Agency World Factbook. <https://www.cia.gov/library/publications/the-world-factbook/>, accessed 26th February 2019.
- CIESIN. 2015. Gridded Population of the World, Version 4 (GPWv4.11), Center for International Earth Science Information Network. <http://www.ciesin.org/>, accessed 11th February 2019.
- Dicker, R., F. Coronado, D. Koo, and R. Parrish. 2011. Principles of Epidemiology in Public Health Practice. United States Centers for Disease Control and Prevention, 84. <https://www.cdc.gov/csels/dsepd/ss1978/SS1978.pdf>, accessed 23rd June 2019.
- FAA. 2018. Aircraft Characteristics Database (last modified 5th October 2018). Federal Aviation Agency (FAA). https://www.faa.gov/airports/engineering/aircraft_char_database/, accessed 26th February 2019.
- FSF. 2019. What is Free Software? Free Software Foundation. <https://www.gnu.org/philosophy/free-sw.en.html>, accessed 10th March 2019.
- Healthline. 2016. How are Diseases Transmitted? <https://www.healthline.com/health/disease-transmission>, accessed 12th April 2019.
- Kelso, N. V. 2018. Natural Earth Raster and Vector Data, Admin 0 – Countries, V4.10. <https://www.naturalearthdata.com/downloads/110m-cultural-vectors/>, accessed 11th February 2019.
- Nandwani, K., M. Wenzel, K. Brockschmidt, and D. Tivel. 2018. Quickstart: Install and use a package in Visual Studio. <https://docs.microsoft.com/en-us/nuget/quickstart/install-and-use-a-package-in-visual-studio>, accessed 11th April 2019.
- Noulas, A., S. Scellato R. Lambiotte, M. Pontil, and C. Mascolo. 2012. "A Tale of Many Cities: Universal Patterns in Human Urban Mobility". *PLOS ONE* 7(5):e37027.
- Patokallio, J. 2018. OpenFlights: Airport, Airline and Route Data. Accessed through Git Hub. <https://github.com/jpatokal/openflights/blob/master/data/airports.dat>, accessed 26th February 2019.
- Salama, R. 2005. "Principles of Communicable Diseases Epidemiology". <https://www.pitt.edu/~super7/32011-33001/32321.ppt>, accessed 9th April 2019.
- Smith, D and L. Moore. 2004. The SIR Model for Spread of Disease. <https://www.maa.org/press/periodicals/loci/joma/the-sir-model-for-spread-of-disease-introduction>, accessed 10th February 2019.
- Volterra, V. 1928. "Variations and Fluctuations of the Number of Individuals in Animal Species Living Together". *ICES Journal of Marine Science* 3(1):3–51.
- Warren, G., G. Hogenson, C. Saisang, S. Maqiv, M. Jones, M. Blome, A. Homer. 2016. Code Generation and T4 Text Templates. <https://docs.microsoft.com/en-us/visualstudio/modeling/code-generation-and-t4-text-templates>, accessed 9th March 2019.
- World Bank. 2017. Net Official Development Assistance and Official Aid Received (Current US\$). World Bank Group. <https://data.worldbank.org/indicator/dt.oda.alld.cd>, accessed 25th February 2019.

AUTHOR BIOGRAPHY

PETER C. BOSCH is a founder of Highpoint Software Systems, a small decision-support technology firm in the upper Midwest. He holds a BSEE from the State University of New York, and is a Certified Java Developer and Microsoft Certified Solution Developer. Pete has published numerous technical articles on object oriented development in these environments, and a half-dozen or so papers on simulation and modeling. He has been designing and building simulations since 1991 for Fortune 100 firms in aerospace, medical imaging, pharmaceutical manufacturing and investment banking, and has been leading large software projects since 1995. He can be reached at Peter@Bosch.Net.