HealthSim: A Management Flight Simulator to Support Resource Collaborative Decision Making for Pandemic Preparedness

Abstract

With the convergence of risk factors driving disease emergence, the amplification and spread of pandemic prone pathogens pose a significant threat to mankind. Public health professionals require tools to help understand the dynamics of disease spread, and the impact resources and collaborative behaviour can have in order to mitigate against potentially adverse outcomes. HealthSim is an interactive, distributed, role-playing management flight simulator, which allows participants to explore key issues related to pandemic preparedness. The system design was informed by a series of workshops with public health professionals, and the implementation is based on open source visualisation and client/server technologies. The paper presents the background and rationale for HealthSim, the underlying spatial simulation model, the system architecture and initial tests and evaluation. Future work is discussed, whereby HealthSim can have a role as (1) as interactive simulator to support public health professionals, and (2) as a experimental laboratory to assess how decision makers perform in dynamic collaborative resource sharing scenarios.

1. Introduction

Decision making in complex systems is challenging, due to factors such as dynamic complexity, time delays between taking a decision and observing its effects, and a lack of awareness of the feedback implications, and side effects, of our actions (J. D. Sterman, 1994). In order to enhance our learning in complex dynamic systems, management flight simulators can provide an immersive learning experience, which encourages reflection and the potential to develop new skills, attributes, and new ways of thinking. Although management flight simulators provide unambiguous feedback about the consequences of diverse sets of policies, this mere interaction does not warrant users to overcome the flaws in their mental models (J. D. Sterman, 1994) . To achieve effective learning, it is necessary to complement the process with a systematic, oriented and enjoyable sequence of activities that allows users to engage in "reflective thought" whereby users not only evaluate policies' performance but their rationale as well. Namely, the management flight simulator needs to be evolved into a *serious game*, where the main purpose is to convey experiential lessons which can be transferred to the real-world system (Lane, 1995).

System dynamics has frequently worked closely with management teams to build transparent and comprehensible simulation models (Lane, 1995), and has a rich tradition of role-playing

flight simulators used to support learning in complex systems, including the beer distribution game (J. Sterman, 1989), fisheries resource game based on tragedy of the commons (Hardin, 1968), world climate role-plays (J. Sterman et al., 2015), US automobile markets (Keith, Naumov, & Sterman, 2017). In the public health sphere, system dynamics models have been successfully deployed to understand the policy options for infectious disease mitigation, from policy models targeting strategies for polio eradication (Thompson & Tebbens, 2008) , to theoretical models of disease spread (Lamberson, 2018). However, to date, there has not been an integrated learning flight simulator to support decision making for pandemic preparedness. This paper describes *HealthSim,* a management flight simulator to support public health professions to explore the impact of an unfolding pandemic, and how their decisions related to resource deployment and sharing can impact the trajectory of an outbreak.

2. Motivation for HealthSim

The context for the development of HealthSim was a collaborative European Union (EU) funded project entitled PANDEM[1], whose purpose was to identify viable innovative concepts to strengthen capacity building for pandemic risk management in the EU (J. Duggan, Hayes, Jilani, Wurmel, & Connolly, 2016). A pandemic is defined as an epidemic occurring worldwide, or over a very wide area, crossing international boundaries and usually affecting a large number of people (Last, Harris, Thuriaux, & Spasoff, 2001). While this definition can be applied to many infectious diseases (e.g. cholera, HIV), the focus of the project was centred on the rapid spread of an infectious agent over a shorter time period, which would place significant stresses on the functioning of health and economic systems. The project was organised across a number of integrated work packages[2], including:

- ***Surveillance***, which involved a comprehensive threat analysis and developed pandemic scenarios to identify gaps. This work also assessed current systems for surveillance and risk assessment at national, EU and global levels to identify good practices for pandemic preparedness and response.

- ***Communications***, which assessed best practice in communications that had previously been used in preparing for, and responding to, earlier epidemics and pandemics.

- ***Governance*****,** which examined the existing legal and policy frameworks at global, European and national levels, with the aim of identifying commonalities, disconnects and priority challenges for future research.

- ***Gaps, Needs and Solutions*****,** which involved workshops to identify needs and innovations to strengthen pandemic management, the development of a matrix to examine current gaps and propose solutions to address these gaps, and the

[1] PANDEM was coordinated by the National University of Ireland Galway, and the partners included: the London School of Hygiene and Tropical Medicine, Université Catholique Louvain, World Health Organisation, Public Health Agency of Sweden (Folkhälsomyndigheten), and the Swedish Defence Research Agency (FOI).
[2] https://cordis.europa.eu/project/rcn/197272/factsheet/en

consolidation of an integrated solution specification to chart the potential of ICT technologies for supporting pandemic preparedness.

Expert workshops formed a key part in the identification and validation of requirements. The first of these was held at the *Metropole Hotel*, Brussels, on February 17-18th, 2016. Participants included the PANDEM consortium and 26 invited experts from the EU and USA from a range of related backgrounds including public health, microbiology, law, defence, security, insurance and crisis management. Three working groups were established: surveillance, communications, and governance. The surveillance working group explored issues such as the identification of pandemic threats, the preparedness phase, the detection phase, and the pandemic phase, where the focus was on good practice, gaps and needs and research and innovations. The communications working group focused on communication between governing bodies and professionals, as well as communication with the general public. It also explored gaps and needs at the policy level (e.g. finding the right level of warning messages to the public), and at a user level (e.g. how to establish trust with highly resistant communities exhibiting "vaccine hesitancy"). The governance working group focused on four priority thematic areas: communication; surveillance; isolation, quarantine, border control and social distancing; and equity and prioritisation of healthcare. Overall, this initial workshop provided a valuable opportunity for the consortium to engage with a wide range of experts, and it led to the identification of a number of key gaps, improvement needs and potential solutions in the three areas of surveillance, communications and governance. This output was then documented, and further input through phone-based interviews with public health experts was conducted, and this information was then made available for the second expert workshop.

The second PANDEM workshop – Integrated Solutions for Pandemic Management - was held at the *Fondation Universitaire* in Brussels on the 21st-22nd September 2016. This workshop systematically reviewed and evaluated the findings from the project's *integrated gap analysis and solution specification*, and engaged with 20 experts from nine EU member states in areas including microbiology, information technology, defence, emergency management and serious games sectors, as well as representatives from the Directorate General of Migration and Home Affairs (DG-HOME), Directorate General for Health and Food Safety (DG-SANTE), and the European Centre for Disease Prevention and Control (ECDC). The workshop took the form of a "living lab" environment, based on a novel influenza virus scenario, which created a common backdrop in which potential new tools, processes and systems could be discussed within a real-world context. Participants were divided into three multi-disciplinary groups, and six topics were presented (informed by the earlier workshop) as research priorities, including: (1) Surveillance and Information Management, (2) Communications (3) Governance, (4) Training and Capacity Building, (5) Logistics, and (6) Diagnostics. A structured discussion was facilitated for each of these areas through a summary that included a description of current gaps, possible solutions, potential impact and ideal situation within 5 years. For example, table 1 shows a summary from topic (1), surveillance and information management, and the overall analysis of gaps and solutions for new predictive modelling tools to assist as part of a pandemic response.

Current Gaps	Analysis of possible development of an epidemic is based on information from traditional sources (sentinel, laboratory). Predictive information is often missing or showing a wide variation of possible outcomes.
Possible Solutions	Transdisciplinary collaborative research used to identify multiple data sources to produce likely scenarios for unfolding pandemic/high impact epidemic.
Potential Impact	Provision of information to implement countermeasures in an efficient way during a pandemic, and flatten the epidemic curve. Give a baseline to enable identification of the most efficient measures.
Ideal Solution within 5-10 years	Predictive analytics that can quickly deliver useful predictive data during an outbreak. Informed by "big data" encompassing a wide spectrum of information feeds, including demographics, environmental data, vector data, individual and traditional data sources.

Table 1: Predictive modelling tool incorporating One Health perspectives

While the workshop and subsequent recommendations focused on six research priorities, the theme that focused on training and capacity building formed the motivating context for the HealthSim solution. Under topic four, it was acknowledged that pandemic management depends on core capacities to prepare and respond to infectious disease threats, and that a knowledgeable skilled workforce is essential. The key role of training, game-based learning, and cross sectoral networking and simulation exercises for pandemic readiness was emphasised. As part of this, a priority challenge was to develop, test and validate a game based learning tool for pandemic preparedness, aimed at pandemic responders and decision makers, where the game replicated the transmission dynamics of a fast moving pathogen, and allowed for the dispensing of resources to mitigate the progression (i.e. "flatten the curve"). The overall process for the HealthSim project is shown in figure 1, where the feedback from expert workshops, literature reviews, and from the design of a resource-based simulator informed the overall design.

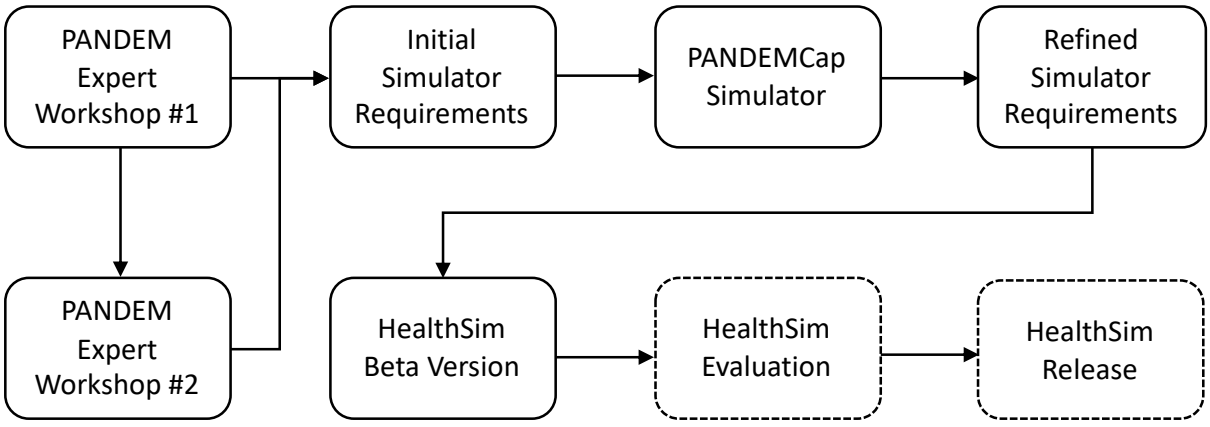


Figure 1: The process of identifying requirements for HealthSim

The initial requirements were further validated through a proof of concept interactive simulator (Yañez, Duggan, Hayes, Jilani, & Connolly, 2017), which built on existing resource-based models for pandemic preparedness (Stein et al., 2012). These requirements encompass a number of features for the flight simulator, namely:

- The game should support public health training aimed at an interdisciplinary team of pandemic responders and decision makers.
- The game should be based on a scenario of a rapid outbreak of a virus, most likely to be a novel form of influenza.
- The game should be team-based, with different roles taken on by each member of a multidisciplinary team.
- The game should record all decisions by participants, in order to facilitate a thorough debriefing session to discuss the decision making logic of each team.
- The game should be based on a spatial transmission model, with a mechanism to model the spread of the virus from one area to another.
- The game should have resource constraints which impact on the burden of disease. For influenza, resources required included vaccines, antivirals and ventilators.
- The game should allow for teams to share resources, in order to counteract the pathogen.
- The game should provide a mechanism to separate those who are severely ill from those who have a mild or moderate infection.
- The game should have an economic cost measure to allow for determining the success of the interventions for each team.

These requirements have formed the basis for the system design and implementation of HealthSim, which is shown as the *beta version* in figure 1. This version, which is close to completion and will be described in more detail in the following sections, will be evaluated on a test group, and feedback will also be gained from public health professionals in relation to the learning outcomes, and the ease of use of the software. Based on these information gathering processes, final changes to the software will be made, and the first release made available. The multiplayer flight simulator will also be supported by comprehensive documentation, similar to the LearningEdge set of simulators (J. Sterman, 2014), including (1) a description of the strategic issue addressed, and the problem area focus, (2) an accompanying case study, which will include a pandemic scenario (E. Vynnycky & Edmunds, 2008), and a summary of how resource modelling can be integrated with transmission models (Stein et al., 2012), and (3) the simulators application areas in terms of training (i.e. public health professionals), and courses, which would include health sciences, resource economics, decision making, and system dynamics.

3. Pandemic Model

The overall structure of the pandemic model is shown in figure 2, and it consists of a number of interrelated model sectors, including: (1) a transmission model, which is based on the SIR model of disease transmission; (2) a number of resource acquisition models that capture decision ordering decisions and also the supply chain dynamics of orders flowing through the different stages, and, (3) a financial model that represents the monetary flows through a country.

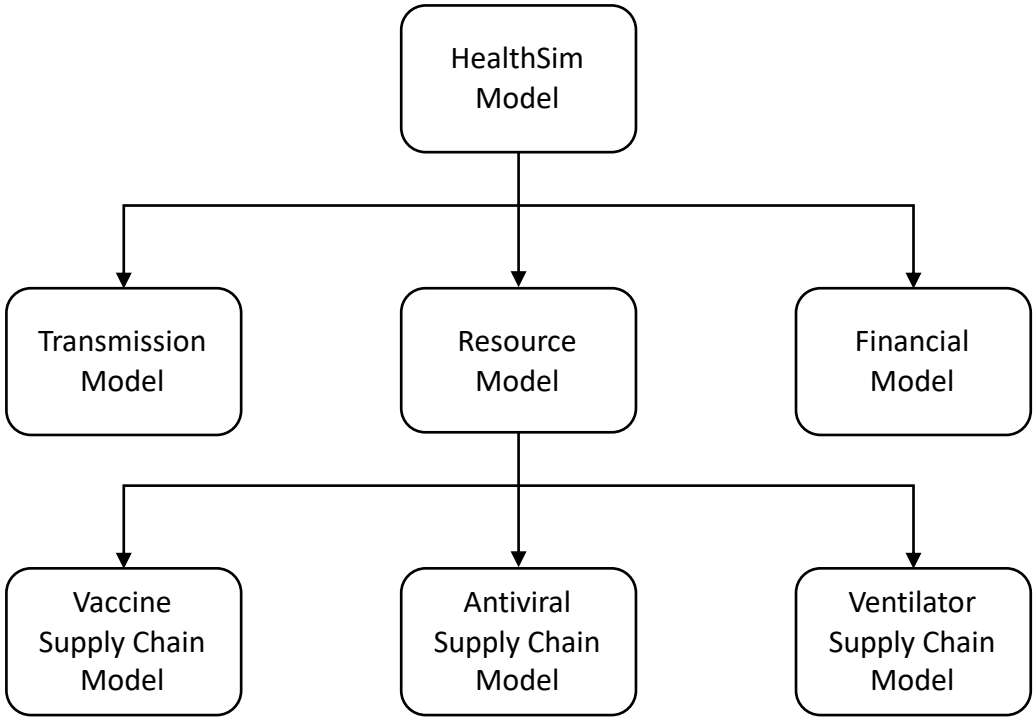


Figure 2: Overall structure of the pandemic preparedness model

This model structure is replicated for each country in the simulation, and each country is allocated a spatial location in terms of an *x-location* and *y-location,* as this information can be used to model infectious contacts between different areas. The transmission model is shown in figure 3, and is an extension of the classic *Susceptibe-Infected-Recovered* (SIR) model from epidemiology (Anderson & May, 1992), with a number of infectious compartments to cover the different scenarios and medical outcomes. These include: *Infected Non-Severe, Infected in Quarantine, Infected Severe* and *Infected Antivirals.* All of these stocks contribute to the calculation of the force of infection for each region, where the force of infection, lambda (λ), is defined as the rate at which susceptible individuals become infected per unit time (Emilia Vynnycky & White, 2010).

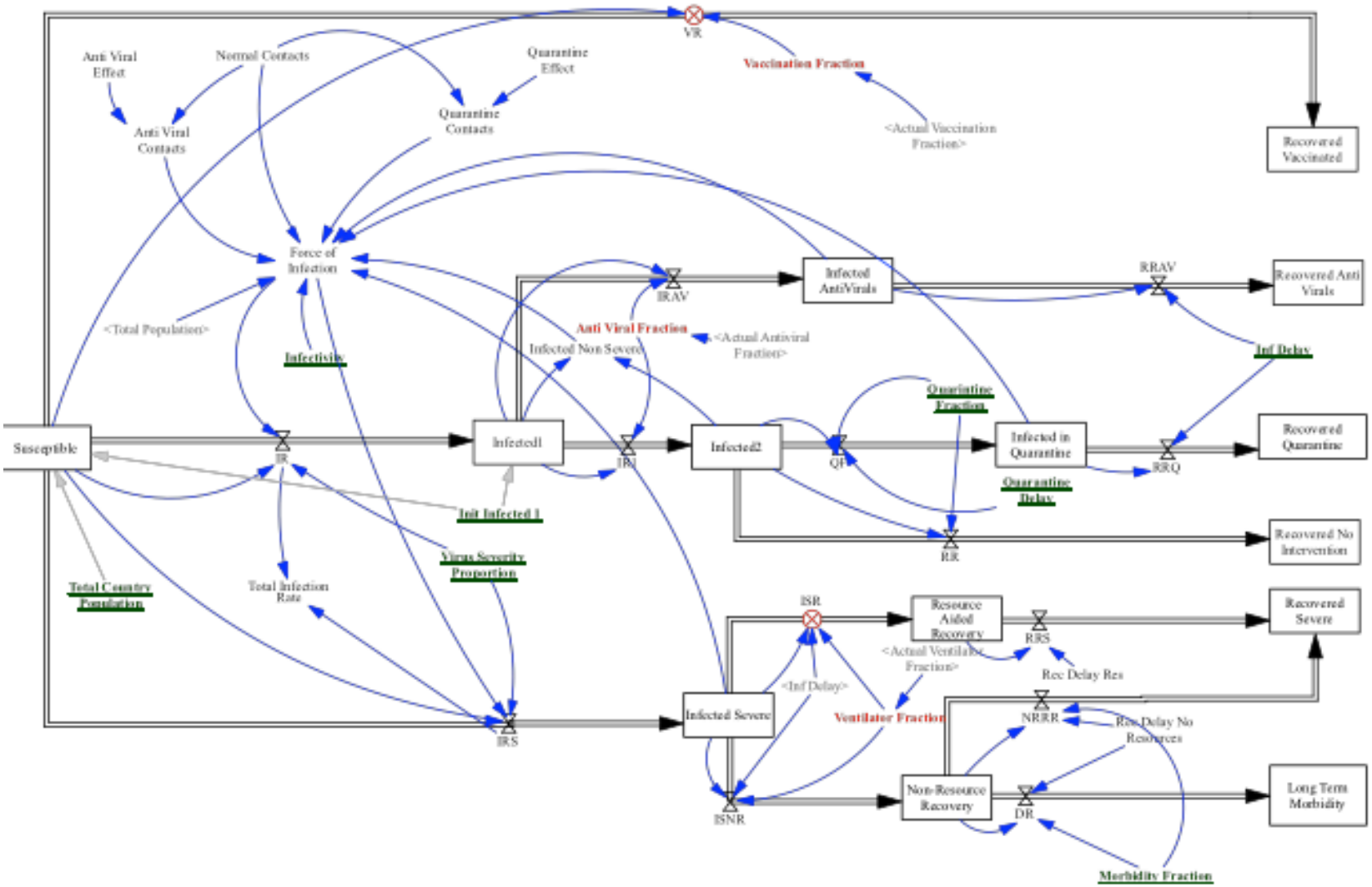


Figure 3: Transmission Model

The spatial contact structure for the model is based on the *whom acquires infection from whom* (WAIFM) matrix structure (E. Vynnycky & Edmunds, 2008), and is captured by the parameter $\beta_{ij,}$ which represents the effective per capita contacts between infectious individuals in region *j* with susceptible individuals in region *i.* These values are moderated by a distancing factor, so that the effective contacts depend on the proximity of countries (1), where a power law equation is used to dampen the effective contact rates, based on the euclidean distance between the two regions (2), where $\alpha \geq 0$.

$$ce_{ij} = ce_{ii} \times \left(d_{ij} + 1\right)^{-\alpha} \tag{1}$$

$$d_{ij} = \sqrt{\left(x_i - x_j\right)^2 + \left(y_i - y_j\right)^2} \tag{2}$$

The force of infection (*λ*) for each region involves a weighted sum of each contributing region (3), for each of the infectious cohorts (*I, IQ, IAV and IS)* specified. The WAIFW matrix (4) is evaluated based on the effective contact matrix from (1), and the population of the susceptible region. A weighting factor $w_i$is used as a multiplier to moderate the contact rates for each infectious group, which could be based on reduced interactions (e.g. patients in hospital who are isolated, or patients remaining at home because of quarantine), or based on a reduced viral load, in the case of patients taking antivirals.

$$\begin{bmatrix}\lambda_1 \\ \vdots \\ \lambda_n\end{bmatrix} = \begin{bmatrix}\beta_{11} & \dots & \beta_{1n} \\ \vdots & \vdots & \vdots \\ \beta_{1n} & \dots & \beta_{nn}\end{bmatrix} \times \left(w_1 \begin{bmatrix}I_1 \\ \vdots \\ I_n\end{bmatrix} + \; w_2 \begin{bmatrix}IQ_1 \\ \vdots \\ IQ_n\end{bmatrix} + w_3 \begin{bmatrix}IAV_1 \\ \vdots \\ IAV_n\end{bmatrix} + w_4 \begin{bmatrix}IS_1 \\ \vdots \\ IS_n\end{bmatrix}\right) \tag{3}$$

$$\beta_{ij} = \frac{ce_{ij}}{N_i} \tag{4}$$

In addition to the infection rate flows (IR and IRS), there are two other flow types in the transmission model:

- *First order delays* that model recovery from infection, and eventually flow into the recovered set of stocks (including long term morbidity for those who are severely infected and do not receive enhanced medical service due to lack of resource availability.
- *Resource constrained flows* (e.g. VR, IRAV, ISR) which are governed by resource availability. In turn resources acquisition is constrained by financial resources, and the supply line delays governing shipments.

The resource constrained flows are specified in the supply chain sub-models, one of which is shown in figure 4. The model is based on the stock management structure (J. Sterman, 2000), but is simplified as the ordering heuristic is exogenous as it will be determined by each team in the flight simulator game. Player orders flow into the supply line, which then are delivered using a pipeline delay (to support a realistic decision environment for players where it would be expected that all orders will arrive together). Players also control a number of additional flows, including antivirals dispensed to infected patients, and antivirals shared with other countries. Donations also change the antiviral stockpile, and a spoilage fraction also reduces the amount of resources available.

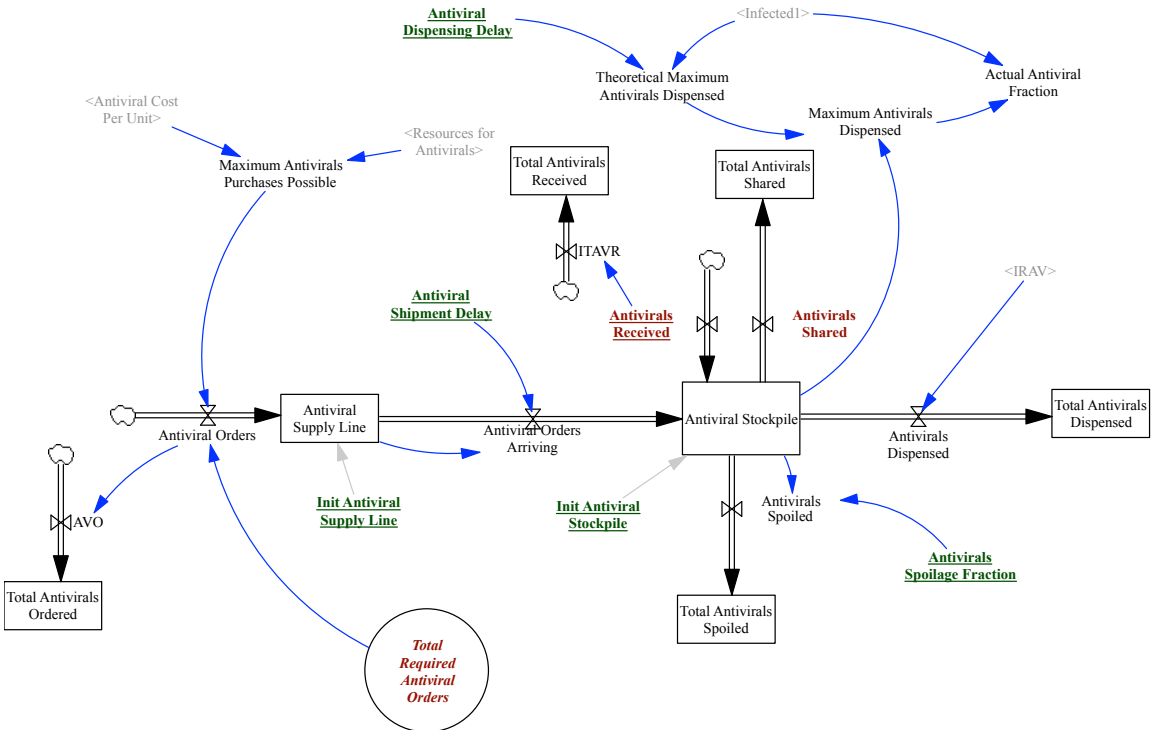


Figure 4: Resource Model

The fraction of patients treated with antivirals is formulated using a first order control structure, expressed in a fractional decrease variable. This was done to simplify the transmission model, and delegate the calculation to the resources submodel. The theoretical maximum resources dispensed is a first order structure (5), and the maximum is then expressed using the min function (6). Finally, this is reformulated as a fractional decrease rate through (7).

$$Theoretical\ Maximum\ Antivirals\ Dispensed = Infected1 / Antiviral\ Dispensing\ Delay \quad (5)$$

$$Maximum\ Antivirals\ Dispensed = \min(Theoretical\ Maximum\ Antivirals\ Dispensed, Antiviral\ Stockpile) \quad (6)$$

$$Actual\ Antiviral\ Fraction = zidz(Maximum\ Antivirals\ Dispensed, Infected1) \quad (7)$$

The third sector type is the financial model, shown in figure 5, which is a simple stock and flow model to keep track of financial resource flows. There is one inflow to the main stock (Financial Resources), and this is donations received from other countries. The outflows include resources donated (to other countries), and the stock is also depleted by spends on the three types of resources (vaccines, antivirals and ventilators). In the game user interface, players are not permitted to spend more than they have accumulated, therefore a mechanism for first order control is maintained for financial resources throughout the game.

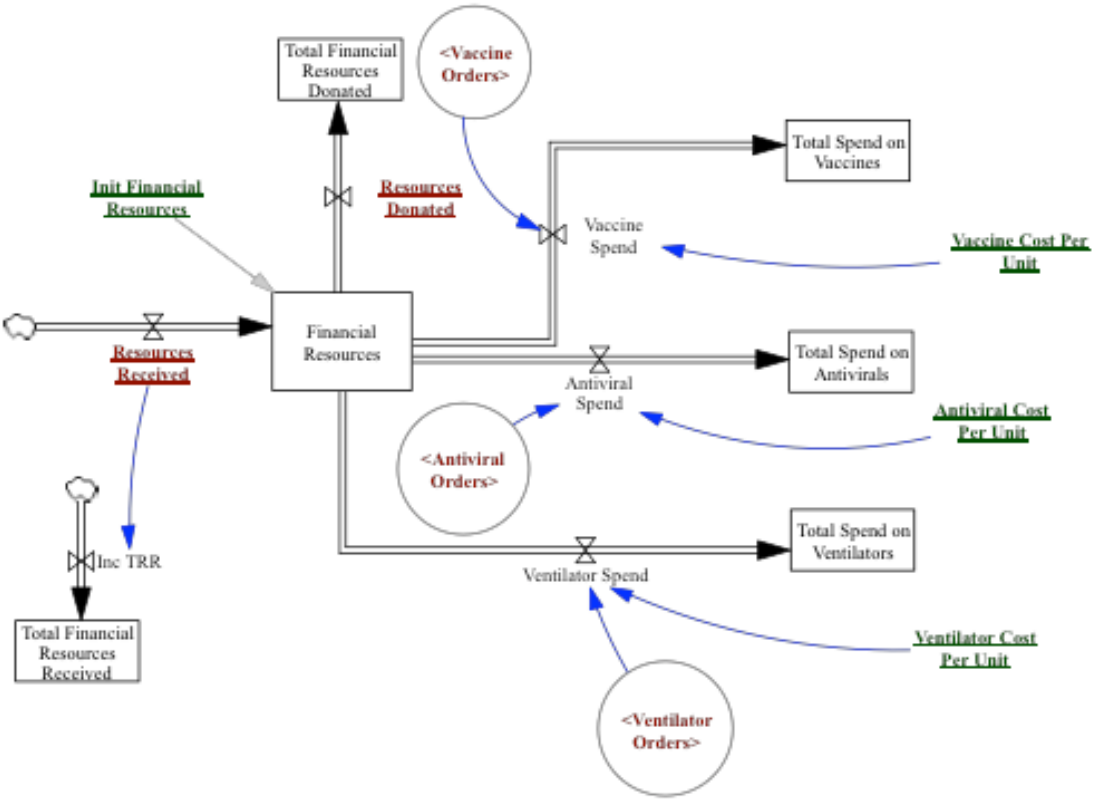


Figure 5: Financial Model

An additional set of equations are required to calculate how well a country performs during the simulation run. This economic measure is based on the stocks in the transmission model that represent those in the population who are unable to work due to illness, or have a reduced capacity for work, because of the long term impact of a severe illness. The equation for cumulative days lost is represented as an integral (8), with the flow a sum of the stocks that impact worker availability (9, 10).

$$Cumulative\ Days\ Lost = INTEGRAL(Days\ Lost, 0) \tag{8}$$

$$\begin{aligned} Days\ Lost = {} & Infected\ NonSevere + Infected\ AntiVirals \\ & + Infected\ in\ Quarantine + Infected\ Severe \\ & + Resource\ Aided\ Recovery + Non\ Resource\ Recovery \\ & + Long\ Term\ Morbidity\ \times\ Days\ Lost\ Fraction \end{aligned} \tag{9}$$

$$Days\ Lost\ Fraction = 0.75 \tag{10}$$

A cost is assigned to the cumulative days lost (11), through a multiplier for average worker productivity. The total cost then uses (11), and factors in all spend on resources (vaccines, antivirals and ventilators), total donations in terms of financial resources, and also any financial resources received. This ensures a *zero sum game* dynamic, where donations will increase the cost base of the donor, and reduce the cost calculation for the recipient.

$$Cost\ of\ Days\ Lost = Cumulative\ Days\ Lost\ \times \\ Average\ Worker\ Productivity \tag{11}$$

$$Total\ Cost = Cost\ of\ Days\ Lost + Total\ Financial\ Resources\ Donated + \\ Total\ Spend\ on\ Vaccines + Total\ Spend\ on\ Antivirals + \\ Total\ Spend\ on\ Ventilators - Total\ Financal\ Resources\ Received \tag{12}$$

4. Software Design Process

As discussed in the introduction section, although management flight simulators provide unambiguous feedback about the consequences of diverse sets of policies, this mere interaction does not warrant users to overcome the flaws in their mental models (J. D. Sterman, 1994). To achieve effective learning, it is necessary to complement the process with a systematic, oriented and enjoyable sequence of activities that allows users to engage in "reflective thought" whereby users not only evaluate policies' performance but their rationale as well. Namely, the management flight needs to be evolved into a 'serious game'.

A serious game is an entertaining game whose main purpose is to convey experiential lessons which can be transferred to the real-world system (Lane, 1995). The implementation of this kind of games may be done with or without the aid of computers. In fact, there exists a board version of the well-known Beer Game which involves virtual crates of beer being moved around a board (J. Sterman, 1989). Notwithstanding this flexibility, the implementation of a game that couples pandemic dynamics with supply chain management renders the task of implementing a physical game impractical. It is even more inappropriate given the number of technological resources available. As a consequence, serious game development is essentially a software development activity.

Because serious games are intended for use by someone apart from the developers (or modellers), crafting these applications require more than writing code haphazardly or creating self-validated interfaces product of the amalgamation of a few time-series, knobs and switches. On the contrary, it involves a structured and step-by-step process to address key aspects of personal software (Sommerville, 2015), including:

- **Acceptability**: Software should deliver the product envisioned by stakeholders.
- **Dependable and security:** Software should not cause physical or economic damage in the event of failure and has to be secure so that malicious users cannot access or damage the system.
- **Efficiency:** Software should not make wasteful use of system resources.

- **Maintainability:** Software should be written in such a way that it can evolve to meet the changing needs of customers.

Software engineering is the discipline that is concerned with these aspects from the early stages of system specification through to maintaining the system after it has been implemented. By adopting a systematic and organised approach, quality software can be produced within schedule and budget (Sommerville, 2015). It has to be noted however, there is no one-size-fits-all method for software development. It is the nature of the expected product and the specific context that determine the appropriate approach to tackle the development endeavour.

It is commonplace that at the start of any project, stakeholders do not clearly envision the goals to achieve, or that halfway down the road, priorities vary due to the ever-changing environment in which they operate. Fortunately, unlike civil engineering where all activities must be thoroughly planned and specified prior to the actual building, software construction permits incremental development through prototyping. By doing so, it is cheaper and easier to make changes in the software as it is being developed (Sommerville, 2015). Any method that adapts swiftly to changes in the environment is known as Agile (Beck et al., 2001).

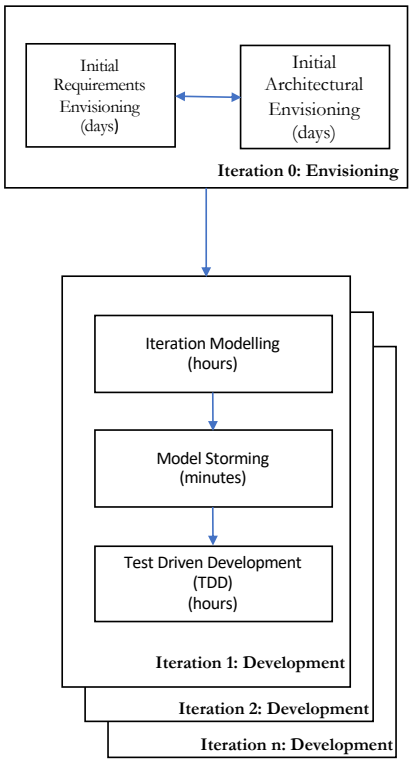


Figure 6. Activities in AMDD

Amongst the many Agile methodologies, the Agile Model Driven Development (AMDD) is deemed pertinent for the management flight simulator development. AMDD focuses on early identification of what the game will do and what technology will support the game development and implementation. This timely envisioning allows the development team to

foresee technical risks, allocate efforts efficiently and engage stakeholders throughout the process.

Figure 6 (Ambler, 2007) depicts the sequence of activities throughout the lifecycle of a project. An agile project begins with an initial vision of the product which subsequently takes shape into a working application through several iterations. At the onset, stakeholders meet to identify the project's high-level scope, draw initial requirements and outline the technology that fits such requirements. This initial envisioning leads the development team to set a task backlog and complete it (sprint) in a period of 1-2 weeks. After this sprint, stakeholders and the development team gather again to observe the produced results. Based on such output, the product vision is refined which translates into additions or removals to the task backlog. Sprints are repeated until the software satisfies or fits its intended use (fulfils requirements – software validation-).

In System Dynamics terms, shown in figure 7, agile development could be represented as a goal-seeking structure. In the absence of floating goals (fluctuations in the 'software vision' stock), the behaviour is dominated by the simple balancing loop on the right (B1), and the software built matches the initial envisioning. However, that scenario is all but unrealistic. As it has been stated, stakeholders continuously modify their expectations and requirements, resulting in increases or decreases to the task backlog (B2 and B3). Consequently, the developer team modifies the software to adjust to these new demands. In other words, the gap between the updated state of expectations and the actual development is dynamically closed.

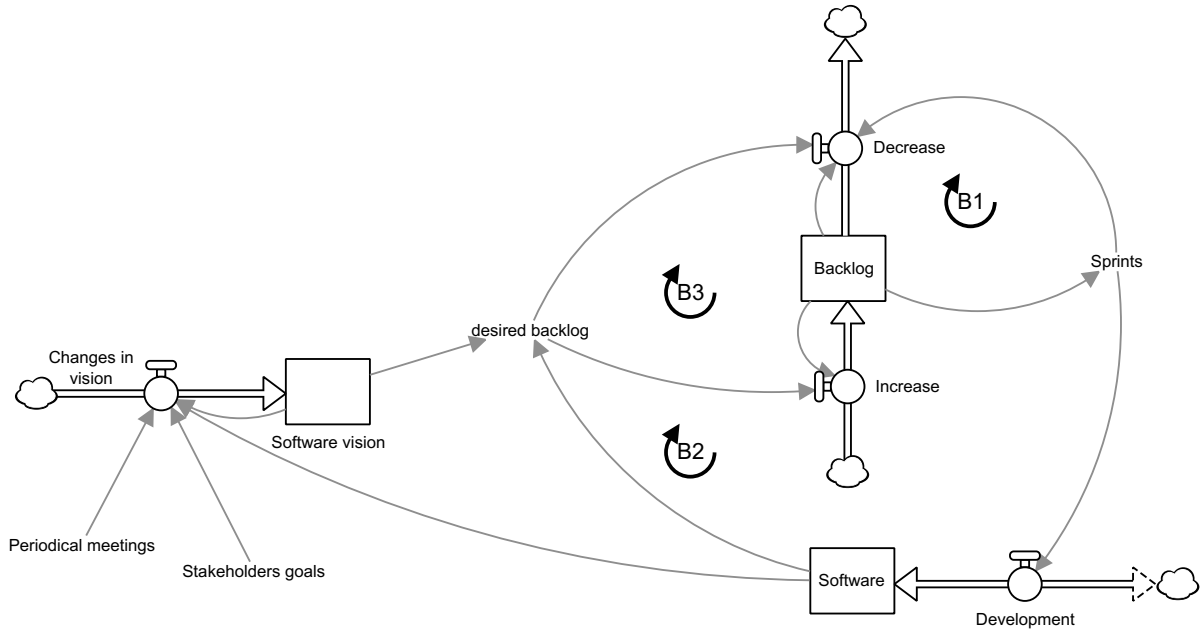


Figure 7. Qualitative diagram of Agile development

5. Functional requirements

According to AMDD, building an application may take several iterations. Implementing HealthSim was no exception, and it demanded various adjustments that range from high-level scope changes to shifts in technological components, in this paper the crafting is presented as a single iteration to illustrate the output from each procedure.

Bearing in mind the goal set for the application – to facilitate the understating of key issues related to pandemic preparedness -, the initial task of any software project is to envision what the product will do to accomplished the defined goal or in software jargon, specify functional requirements. Use case methodology fits properly in this task due to it serves to communicate from one person to another, often to people with no special training, the system's behaviour under various conditions as it responds to a request from one of the stakeholders in a simple text or diagram. They are popular largely because they tell coherent stories about how the system will behave in use (Cockburn, 2000).

In this case, HealthSim interacts primarily with two stakeholders: an instructor and a set of players (Figure 8). Both need to validate their credentials to gain access to the system's functionalities. Instructors create sessions by the configuring game's conditions and teams' characteristics. Once sessions have been configured, players join teams. After both actors complete the game setting, the system displays a tailored interface to each individual depending on the role and team (if he or she is a player). In these interfaces, players make decisions to fix a problematic situation. Subsequently, the instructor collects players' policies to feed them as an input to the system dynamics model. Afterward, the application simulates the model and presents the results on the interfaces. Moreover, while players deliberate their course of action, they interact with fellow users via chat.

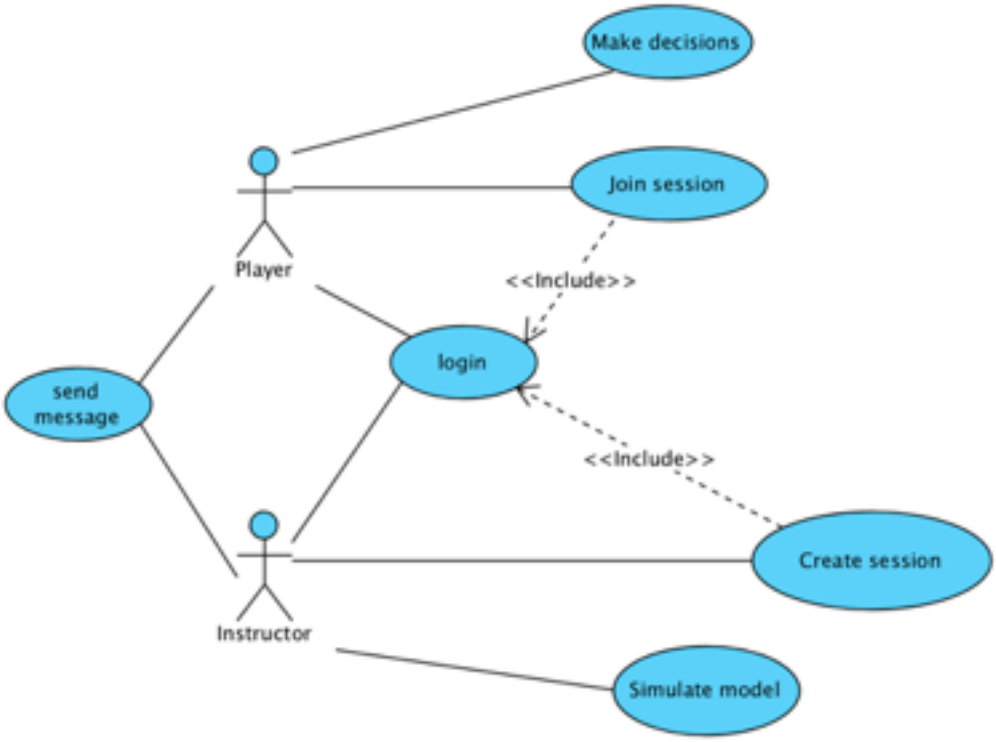


Figure 8. Use case diagram

6. Non-functional requirements

Thus far, the main requirements do not yet impose technological restrictions. Any tool that fulfils these specifications is a candidate to serve as the implementation means. For making such a crucial choice, it is imperative to think about the "quality attributes" of the software. In other words, how each use case is executed. These attributes are also known as non-functional requirements.

SD Simulation

As it has been stated, users are intended to recognise the complexity in pandemic risk and emergency management through the interaction with System Dynamics models. It follows that the software must run these microworlds. However, the constraint is not limited only to run simulations; the application must allow 'discrete steps' to feed the model with inputs from the players.

Back-end server

The game in its simplest form consists of an instructor and a player who share information between themselves. In a more regular setup, the number of players may exceed half a dozen and each player makes more than one decision per round. This situation entails a need for the instructor to collect tens of inputs in just one round. Without computer aid, the instructor would manually organise and transform all the information pieces in the specific format required by the application. Besides being a cumbersome task, any potential mistype jeopardises the integrity of the results. To avoid such a pitfall, the application must automatically collect data from the players, format the information and feed it into the simulation model. Therefore, the application must include a communication system working through several workstations supported by a server or back-end application.

Web

In addition to the gameplay, the setup stage must be as seamless as possible. Since the game ought to be an enjoyable experience, lengthy software installations may frustrate players and decrease their interest in the exercise. Hence, the application must require little to none installation procedures. Notwithstanding that the variety of operating systems (Windows, Mac, Linux, Android, iOS) and devices (laptop, desktop, smartphone, tablet) put an apparent strain on application's adaptability; thanks to web-standard technologies, applications can be used by anyone through a recent web browser.

Visual design

Amongst the reasons (Sterman, 1994; Lane, 1995) that hinder microworlds' effectiveness, visual design is frequently overlooked or considered merely in an aesthetic fashion, rather than a feature that may obscure the insights from a useful model by communication shortcomings. In serious games, developers strive for mimicking real-life dashboards that inform decisions. A dashboard is a visual vehicle that conveys the most important information needed to achieve one or more objectives. It is often consolidated on a single computer

screen so it can be monitored at glance. Surprisingly, most information dashboards that are used in organisations fall far short in their potential (Few, 2013). Few also contends that:

"*The root of the problem is not technology, at least not primarily – but poor visual design. To serve their purpose and fulfill their potential, dashboards must display a dense array of information in a small amount of space and in a manner that communicates clearly and immediately. This requires design that taps into and leverages the power of visual perception, which enables us to sense and process large chunks of information rapidly. This power can only be utilized when the visual design of dashboards is central to the development process and is informed by a solid understanding of visual perception*."

In short, visual design involves a thoughtful reflection on what works, and why, regarding perception. Such an understanding facilitates the communication of dynamic behaviours through small, concise, direct and clear display media. Moreover, to serve its purpose, dashboards must be adapted to the requirements of a person or a group of people (language, conventions, and so on). Taking into account that effective communication and customisation play important roles in the application's success, it thus follows that the development tool must support the creation of flexible data visualisations grounded on sounded visual design principles.

Databases

Serious games' learning opportunities are not exclusive to players; each game session provides rich amounts of data from which developers and modellers can enhance the product (software and the underlying SD model). To facilitate this process, the application must store the information generated from players' decisions information in one or more databases.

Version control

The process of developing an application should be an enjoyable experience for developers as well. Given that requirements vary more often than not, during the development phase and even after the implementation phase, each component is modified, dropped, or even recovered due to stakeholders' second thoughts. These fluctuations may wreak havoc on the building process should an appropriate version control method is not integrated. Version control means the management of any changes to the application. A well-functioning version control system allows a developer to keep track of every single change in an application since its inception and restore previous versions seamlessly. Nevertheless, change management should mirror on every developer's computer the complete codebase - including its full history. Hence, the application must support distributed version control.

Code

A corollary that stems naturally from the previous requirement is that the application must be built entirely in programming code, instead of point-and-click or Graphical User Interfaces – GUI-. Besides enabling version control, well-written code is reproducible: it describes unambiguously each step that produces a certain result.

Furthermore, GUIs do not provide the necessary flexibility to tailor an application to the distinct challenges that each effort demands. On the contrary, pre-built components delimit what the software can and cannot do (as any good software). However, serious game development requires an ample and eclectic set of tools that a single GUI cannot provide. In some situations, those tools may not even exist. Therefore, the creation of new functionalities is almost inescapable.

Open-source

At the beginning of this section, it was mentioned that any tool that meets the functional requirements could serve as the implementation tool. After introducing non-functional requirements the number is significantly reduced, to none. There is indeed no single proprietary development software that adheres to such a description. As a consequence, one must resort to a quest of components that perform one or various requirements, and equally important, that seamlessly integrate. Yet again, commercial software's cost precludes this course of action.

Fortunately, during the last decades, there has been an explosive growth of open-source technology, ranging from interface design to esoteric databases. This type of software allows any person to inspect, modify, and enhance the source code. It is not surprising that many companies nowadays build their businesses around this community-based and free[3] technology. In addition to lower costs, open source software benefits development by preventing the chaos generated by vendors when they stop producing or supporting a product. Source code availability fosters a culture of collaboration, offers a great degree of customisation, exposes vulnerabilities for all to see, to the extent that users may fix errors or even improve functionalities by themselves. These features have shaped a rich and variegated ecosystem of specialised tools that share communication standards, resulting in an unbounded development catalogue from which programmers may choose.

[3] The term "free" is conceived in terms of accessibility, not in terms of price. However, almost all free software is gratis.

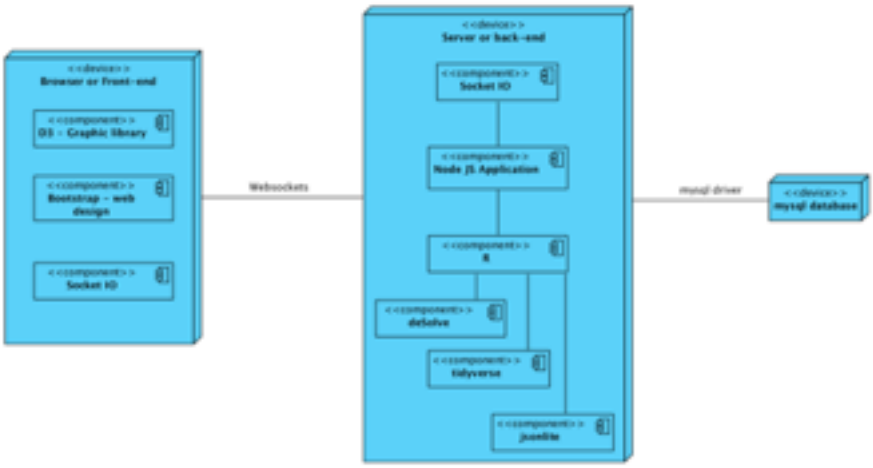


Figure 8. Architecture diagram

The architectural envisioning is consolidated by conflating all non-functional requirements into a diagram (figure 8) that describes the application's high-level structures and the associated technologies.

On the front end, a user (either player or instructor) interacts with the application through a web-browser interface whose visual design is supported by Bootstrap and D3. On the one hand, Bootstrap is an open source toolkit for developing with HTML, CSS, and Javascript. It boosts web design thanks to its catalogue of pre-built blocks of code. On the other hand, D3 is a JavaScript library for producing dynamic, interactive data visualizations (Murray, 2017), for example:

> *The abbreviation D3 references the tool's full name, Data-Driven Documents. The data is provided by you, and the documents are web-based documents, meaning any- thing that can be rendered by a web browser, such as HTML and SVG. D3 does the driving, in the sense that it connects the data to the documents.*
>
> *Of course, the name also functions as a clever allusion to the network of technologies underlying the tool itself: the W3, or World Wide Web, or, today, simply "the web."*

Hence, by the use of the widely implemented SVG, HTML, and CSS standards, D3 allows great control over the final visual result. This entails an unconstrained design in which effective visual principles can be applied.

On the back end, a server component handles requests from users sent through a bi-directional communication channel (WebSocket) created by Socket IO, a JavaScript library. Each request consists of an instruction and an optional payload. Based on the instruction, the server component performs a specific process to meet users' demands. Such a component is implemented in Node JS, an open source development platform for executing JavaScript code server-side. Node is often used for real-time applications given that it provides a persistent connection from the browser to the server.

Although Node itself provides a wide array of capabilities, it does not include model simulation. It can, however, assign such a task to a more competent delegate and act as an intermediary between users and that delegate: R. It is a programming language and free software environment for statistical computing and graphics. Due to the vast community of developers which have built over 10.000 libraries, System Dynamics simulation is part of R's toolkit (Jim Duggan, 2018) thanks to the deSolve library. In conjunction with the packages Tidyverse and Jsonlite, results can navigate all the way from the server through the user's browser given that all the parts along the streamline share a communication standard (JSON).

Finally, Git and Github work in tandem to manage changes in the codebase. By doing so, it is possible to compare files, identify differences, and merge the changes if needed prior to committing any code. Moreover, it allows developers to keep track of application builds by being able to identify which version is currently in development, testing, and production. Likewise, when new developers join the team, they can easily download the current version of the application. During development, if necessary, developers may work in different independent code versions (branches). When ready, Git and GitHub merge those branches to create a final working version.

The synergy of the above open-source components morphs into a framework that facilitates the development of an enjoyable, didactic and serious game experience.

7. The game

Once the requirements have been set out, the ensuing step is to implement such concepts in a working prototype. The game consists of a group of players who join teams to take actions that cope with an infectious outbreak. Each team represents a country and starts with a pre-determined number of resources (antivirals, vaccines, ventilators, and financial resources). In every simulation round, each country must decide the number of resources to deploy within its jurisdiction and how many resources they donate to other countries. The primary goal of the game is to minimise the economic loss.

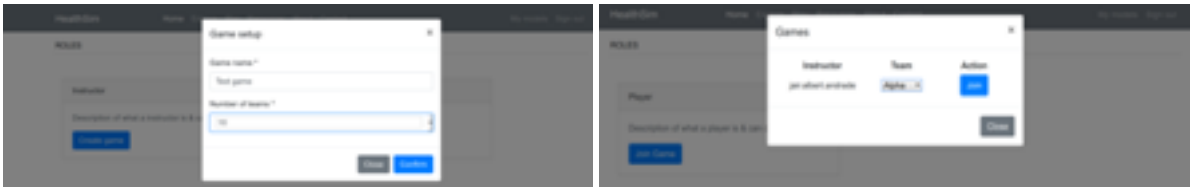
In this section, the developed game is presented through an event sequence, starting from user authentication and ending with the last round of simulation.

i. Login

Figure 9. Social network authentication

On the welcome screen (figure 9), the interface prompts users to log in by means of social networks.

ii. Create/join game



(a) *(b)*

Figure 10. Choose roles

After users have logged in, they may choose one of two courses of action: a) Select the role of ‘instructor’ and create a new game session which other users may join, or b) Select the role of ‘player’, and join a team in a session created by an instructor.

iii. *Instructor: Configure the game*

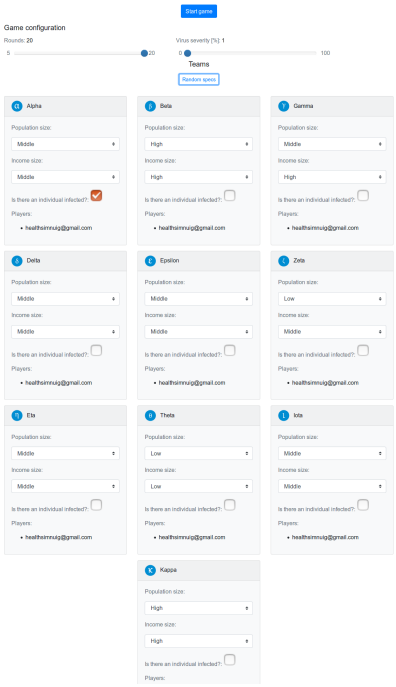


Figure 11. Instructor's setup screen

Following the creation of a game session, the application displays a setup interface to instructors (figure 11). In this screen, instructors can determine the characteristics of each team (population size and income); which teams have infected individuals; the number of rounds; and the severity of the virus. Additionally, the interface notifies the instructor each time a player joins the game.

iv. *Players: Make decisions*

Figure 12. Player's dashboard

On each player screen, a tailored dashboard appears as soon as the instructor starts the gameplay. This dashboard consists of three sections. First, a decision board, in which players decide how many resources deploy within their jurisdiction, resource purchasing, and the number of resources to donate to other teams. Second, an information board that displays team characteristics, key performance indicators, epidemiological curves and resources' behaviour over time (through classical time series and unfamiliar sparklines). Lastly, a chat board, a communication system among players and the instructor.

It is important to note that the information about infected individuals is deliberately lagged to players. The magnitude of such a delay is proportional to the team's income size. The introduction of this feature entails that players make decisions based on delayed pieces of data.

v. Instructor: Simulate and initiate each round

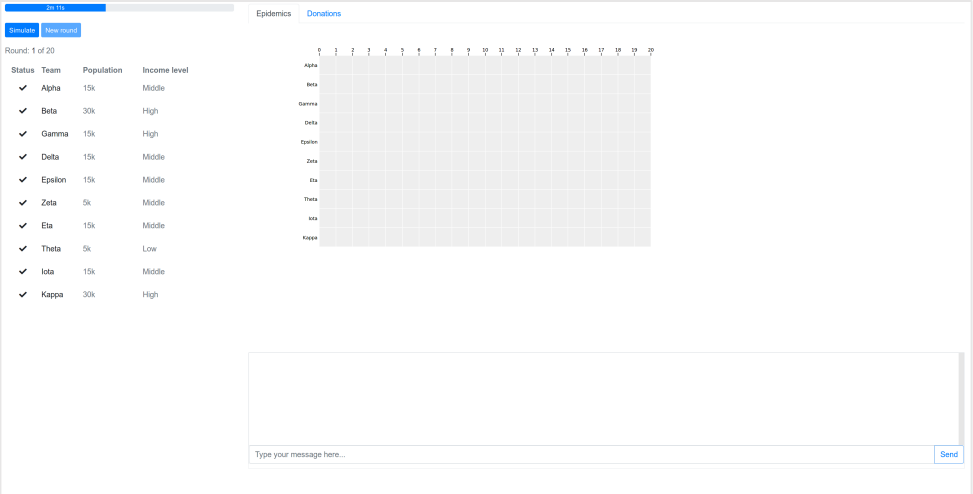


Figure 13. Instructor's dashboard

On the other hand, while players think of strategies, the instructor waits for all players to send their decisions. The application allows the instructor the check whether a team has sent the required information. As well as with players' dashboard, instructor's interface includes a chat, information and decisions boards. The latter consists of two buttons: to simulate the model and to start a new round.

vi. Players: Simulation results

Figure 14. Player results

In each round, the application updates players' dashboard based on the simulation results. Figure 14 shows team Theta's dashboard in round 19.

vii. Instructor: Simulation results

Figure 15. Global results

Likewise, the application updates the instructor's dashboard. From the graphical artefacts (heatmap, chord diagram, and bar chart) on the information board, the instructor can conduct a debrief about the dynamic reasons that lead to the obtained behaviour.

All in all, the result is an open source and cloud-based application in which public health stakeholders can interact with System Dynamics models of infectious diseases through a user-friendly environment. In doing so, stakeholders gain in-depth understanding of the processes that lead to the transmission of infectious diseases and test the likely effects of control strategies in order to identify, design and guide the implementation of policies that ensure

that the right health product arrives for the right person at the right time, an application also known as HealthSim.

8. Conclusions and Next Steps

The usefulness of management flight simulators has long been discussed in the System Dynamics community. The paper presents the background and rationale for HealthSim, the underlying spatial simulation model, the system architecture and initial tests and evaluation. HealthSim's development account shows that crafting a management flight simulator is a significant software software endeavour. Future work will further develop HealthSim can have a role as (1) as interactive simulator to support public health professionals, and (2) as a experimental laboratory to assess how decision makers perform in dynamic collaborative resource sharing scenarios.

9. References

Ambler, S. W. (2007). *Agile software development at scale.* Paper presented at the IFIP Central and East European Conference on Software Engineering Techniques.

Anderson, R. M., & May, R. M. (1992). *Infectious Diseases of Humans: Dynamics and Control*: Oxford University Press.

Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., . . . Jeffries, R. (2001). Manifesto for agile software development.

Cockburn, A. (2000). *Writing effective use cases*: Addison-Wesley Professional.

Duggan, J. (2018). Input and output data analysis for system dynamics modelling using the tidyverse libraries of R. *System Dynamics Review, 34*(3), 438-461. doi:doi:10.1002/sdr.1600

Duggan, J., Hayes, C., Jilani, M., Wurmel, J., & Connolly, M. (2016). A Multisectoral Approach to Identify Innovative Solutions to Strengthen Capacity Building for Pandemic Risk Management. *International Journal of Infectious Diseases, 53*, 112-113. doi:10.1016/j.ijid.2016.11.282

Few, S. (2013). *Information Dashboard Design: Displaying data for at-a-glance monitoring* (Vol. 5): Analytics Press Burlingame, CA.

Hardin, G. (1968). The Tragedy of the Commons. *Science, 162*(3859), 1243-1248. doi:10.1126/science.162.3859.1243

Keith, D. R., Naumov, S., & Sterman, J. (2017). Driving the Future: A Management Flight Simulator of the US Automobile Market. *Simulation & Gaming, 48*(6), 735-769. doi:10.1177/1046878117737807

Lamberson, P. J. (2018). Approximating individual interactions in compartmental system dynamics models. *System Dynamics Review, 34*(1-2), 284-326. doi:doi:10.1002/sdr.1599

Lane, D. C. (1995). On a Resurgence of Management Simulations and Games. *Journal of the Operational Research Society, 46*(5), 604-625. doi:10.1057/jors.1995.86

Last, J. M., Harris, S. S., Thuriaux, M. C., & Spasoff, R. A. (2001). *A dictionary of epidemiology*: International Epidemiological Association, Inc.

Murray, S. (2017). *Interactive data visualization for the web: an introduction to designing with*: " O'Reilly Media, Inc.".

Sommerville, I. (2015). *Software Engineering (10th Edition)*: Pearson.

Stein, M. L., Rudge, J. W., Coker, R., Van Der Weijden, C., Krumkamp, R., Hanvoravongchai, P., . . . Adisasmito, W. (2012). Development of a resource modelling tool to support decision makers in pandemic influenza preparedness: The AsiaFluCap Simulator. *BMC public health, 12*(1), 870.

Sterman, J. (1989). Modeling Managerial Behavior: Misperceptions of Feedback in a Dynamic Decision Making Experiment. *Management Science, 35*(3), 321-339. doi:10.1287/mnsc.35.3.321

Sterman, J. (2000). *Business dynamics: systems thinking and modeling for a complex world*: McGraw-Hill.

Sterman, J. (2014). Interactive web-based simulations for strategy and sustainability: The MIT Sloan LearningEdge management flight simulators, Part I. *System Dynamics Review, 30*(1-2), 89-121. doi:doi:10.1002/sdr.1513

Sterman, J., Franck, T., Fiddaman, T., Jones, A., McCauley, S., Rice, P., . . . Rooney-Varga, J. N. (2015). WORLD CLIMATE:A Role-Play Simulation of Climate Negotiations. *Simulation & Gaming, 46*(3-4), 348-382. doi:10.1177/1046878113514935

Sterman, J. D. (1994). Learning in and about complex systems. *System Dynamics Review, 10*(2-3), 291-330. doi:doi:10.1002/sdr.4260100214

Thompson, K. M., & Tebbens, R. J. D. (2008). Using system dynamics to develop policies that matter: global management of poliomyelitis and beyond. *System Dynamics Review, 24*(4), 433-449. doi:doi:10.1002/sdr.419

Vynnycky, E., & Edmunds, W. J. (2008). Analyses of the 1957 (Asian) influenza pandemic in the United Kingdom and the impact of school closures. . *Epidemiology and infection, 136*(02), 166-179.

Vynnycky, E., & White, R. (2010). *An introduction to infectious disease modelling.* : Oxford University Press.

Yañez, A., Duggan, J., Hayes, C., Jilani, M., & Connolly, M. (2017, 1-1 Oct. 2017). *PandemCap: Decision support tool for epidemic management.* Paper presented at the 2017 IEEE Workshop on Visual Analytics in Healthcare (VAHC).