Modelling Pandemic Influenza Progression Using Spatiotemporal Epidemiological Modeller (STEM)

By

Hui Zhang

B.Eng., Electrical Engineering, National University of Singapore (2008)

Submitted to the School of Engineering

In partial fulfillment of the requirement for the degree of

Master of Science in Computation for Design and Optimization

At the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2009

Author .....

School of Engineering

September 2, 2008

Certified by.....

Richard Larson

Mitsui Professor of Engineering System and Civil and Environmental Engineering

Thesis Supervisor

Accepted by.....

Jaime Peraire

Professor of Aeronautics and Astronautics

Director, Computation for Design and Optimization Program

Modelling Pandemic Influenza Progression Using Spatiotemporal Epidemiological Modeller (STEM)

By

Hui Zhang

B.Eng., Electrical Engineering, National University of Singapore (2008)

Submitted to the School of Engineering
In partial fulfillment of the requirement for the degree of
Master of Science in Computation for Design and Optimization

Abstract

The purpose of this project is to incorporate a Poisson disease model into the Spatiotemporal Epidemiological Modeler (STEM) and visualize the disease spread on Google Earth. It is done through developing a Poisson disease model plug-in using the Eclipse Modeling Framework (EMF), a modeling framework and code generation facility for building tools and other applications based on a structured data model. The project consists of two stages. First, it develops a disease model plug-in of a Poisson disease model of a homogenous population, which is built as an extension of the implemented SI disease model in the STEM. Next, it proposes an algorithm to port a Poisson disease model of a heterogeneous population into the STEM. The development of the two new diseases plug-ins explores the maximum compatibility of the STEM and sets model for potential users to flexibly construct their own disease model for simulation.

Thesis Supervisor: Richard Larson
Title: Mitsui Professor of Engineering System and Civil and Environmental Engineering

Acknowledgements

First of all, I would like to thank Professor Richard Larson for kindly agreeing to be my supervisor, setting up the research scope and topic, his invaluable guidance and generous help during this one year. His wide knowledge in disaster response field, interesting way of teaching and doing research inspired me in many aspects. Working with him and his flu team has been a fun and rewarding experience.

I also want to thank Daniel Ford, James Kaufman and other members from IBM STEM development team for their timely responses to my questions and help for this project.

Next, I would like to thank my friends Yuan Luo and Dheera Venkatraman from MIT EECS for their time, patience in sharing their knowledge and experience in using Eclipse and Java programming.

I would also like to appreciate the great work by Laura Koller from CDO office for administrative assistance, Prof Jaime Peraire for giving advice in my academic work.

Last but not least, I would like to thank all my friends from CDO program, Ashdown Housing and my lovely flat mates for their support and encouragement during this one year. You made my life in MIT fun, memorable and fantastic!

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Scientists and researchers have been doing epidemic disease modeling and simulation for years. In the majority of the research works, they have to develop various software and computer programs for the specific modeling and simulation tasks. Many of these software and computer programs are not suitable for reuse in other tasks. Moreover, many of these simulations incorporate only the disease information on a fixed population size, and they fail to reflect the geographic information of world population [1].

In the last few years, the idea to improve the state-of-the-art disease modeling techniques and use visualization to better understand disease models led to the development of Spatiotemporal Epidemiological Modeler (STEM). STEM is an epidemic disease modeling software program developed by IBM [1] [19]. With data sets for countries defined using the ISO-3166 standard [18], it provides a platform to visualize the progression of an epidemic disease on an electronic map, as depicted in Figure 9 later. It also has a gateway to Google Earth [20], which imposes the disease status and population information onto the map images from Google Earth. Another innovative feature of STEM is its flexibility and extensibility. Researchers can design different modeling scenarios through combining various disease parameters or implement their own disease models and plug them into STEM systems. All these models can be reused in other modeling scenarios or serve as basic models to be enhanced.

STEM is designed to support most types of disease models and has implemented several textbook disease models such as SI, SIR and SEIR [2]. Generally, these standard textbook disease models satisfy the needs for modeling and conducting simulations of various types of diseases. Researchers have to change the only input parameters such as transmission rate and mortality rate according to the purpose of simulation. However, some researchers have developed their own disease models built on the basic features of the standard models. For example, a Poisson disease model has been built which associates the transmission rate with the level of social contacts of individuals within a population [3]. To conduct both deterministic and stochastic simulations for this kind of user defined disease model in STEM, the modeler needs to go through the process of building a disease plug-in through Eclipse Modeling Framework (EMF).

EMF is the platform on which STEM is built. It is a Java modeling framework and code generation facility which helps developers to build tools and applications on a structured model [4]. The functions of STEM are built in the Eclipse before they are embedded into the application as plug-ins. To define new disease models, all modelers go through the same process. First, the modeler needs to understand the structure and design of STEM and work out algorithms to fit their model into this system. Second, the modeler has to code its algorithms in Java and put the code into the predefined location of the system. Third, after the modeler defines all the paths and locations of the edit code and editor code, EMF can help to generate the rest of the code into the correct position in a structured way. Finally, the user needs to plug in the disease model before running STEM on Eclipse platform.

This project report is structured in the following way. First, we review the background of epidemic disease modeling and related literatures. After that, we explore the functions of

creating scenarios using implemented disease models in STEM. We then introduce ways of developing a disease plug-in for the Poisson disease model on a homogenous population. Finally, we propose an algorithm for developing the same disease model on a heterogeneous population. Due to time limitation and lack of documentation, the implementation of these two models is not completed. Some evaluations and suggested improvements of the software will be discussed at the end.

Chapter 2

Literature Review

2.1The background of influenza epidemics

Influenza is an infectious disease which spreads among birds and mammals. Some of the most common symptoms are fever, cough and muscle pain. Although the symptoms are similar to common cold, it is more dangerous than the common old due to the following reasons.

First, due to its infectious nature, it may cause large epidemics among animal or human populations. These epidemics not only take sometimes millions of lives, but also cause anxieties and fears which will lead to indirect or intangible costs in our normal lives. For example, 2009 H1N1 flu was first named as swine flu, which triggered anxiety among people that pork might be poisonous. Some countries like China and Russia banned pork imports from certain states in the US [5]. These cause a decline of pork sales in the U.S and greatly harmed the pork industry.

Second, vaccines are usually not available at the initial stage of pandemic progression. Influenza is caused by various types of virus. For seasonal influenza, vaccines are at stock and ready to be distributed in each flu season. However, there is usually a lag between the vaccine development and the outbreak of the influenza caused by any new virus. The disease becomes uncontrollable using a pharmaceutical approach at the initial stage. Even after the development of the vaccine, the virus might mutate during the phase of

transmission and the vaccine becomes ineffective for the mutated virus, which requires a new round of effort to develop the new vaccine. Developing vaccine is costly both in terms of time and money.

Third, influenza is more deadly than common cold. While the symptoms of common cold are most of the times just sneeze and headaches, influenza will cause fever almost every time. According to historical figures, the 1918 Spanish flu killed 50-100 million people world wide, which is an extreme example of how deadly flu is. Even the seasonal flu kills 36,000 people in the U.S. annually [17]. Common cold rarely kills people, unless it causes some complications which might be deadly.

There were several large influenza pandemics in the history. Each of these enlightened us on the prevention and control of flu pandemics.

In the 1918 Spanish flu, the virus appeared to be mild when the flu first started to spread. However, it mutated into a more fatal virus, caused two more waves of flu pandemic worldwide and killed millions of people [6]. The Spanish flu has a high transmission rate (estimated to be 3.75 for the second wave) and a high mortality rate (estimated to be 10% to 20%). Besides the fatal nature of the virus itself, lack of medication or global cooperation were some of the reasons that caused the human catastrophe. The flu occurred during World War I. Soldiers played a major role as disease carriers and there was no space for cooperation at that certain time.

The 2005 Avian flu has the characteristics of a high mortality rate (up to 60%) but a low transmission rate [7]. Most of the infection cases are from poultry to human beings. Some

evidence shows that the human to human transmission is limited, but endemics among poultry population are hardly evitable in certain areas. Its low transmission rate among human population might be the major reason that a serious global pandemic has not been aroused.

Last but not least, 2009 H1N1 flu tests our preparedness on potential global epidemics. The flu virus so far shows a high transmission rate but a relatively low death rate. Some non-pharmaceutical methods have been taken by authorities and publics such as school closing, frequent hand washing and masks wearing. All these methods proved a certain level of effectiveness.

Influenza epidemic is one of the major issues concerned by scientists and public health authorities. The modelling of influenza progression is helpful for them to understand and control this disease. Scientists have been developing mathematical progression models and conducting simulation for them. Some of the control methods can also be represented mathematically and incorporated into the disease model. The application of these models can help disease modellers and health authorities better understand the disease and evaluate the effectiveness of various control methods.

2.2 Mathematical epidemic models

The progression of epidemic disease, including influenza, can be modelled mathematically using epidemic models. Some of the most extensively used textbook models are implemented into STEM. These models are modified to cater to the spatial nature of STEM modelling.

SI(S) Model [8]

The most basic model is SI(S) model, where S stands for Susceptible and I stands for Infectious. The population is assumed to be homogenous, well-mixed and divided into these two groups. This disease model assumes the population immediately becomes infectious once is infected. Moreover, for SIS model, it assumes people who have been infected before do not have immunity of this disease, and can be infected again in the future. One example of SIS model is the common cold. The transition diagram of this model is as follows. The meanings of the symbols are as follows.

β refers to the transmission rate, the rate that an infectious person would affect and susceptible person. γ is the rate of losing infectiousness, and is 0 in a pure SI model. μ is the immigration rate and emigration rate. They are assumed to be equal such that the total population remains the same. The relations of these variables are as follows.

$$\Delta s = \mu - \beta si + \gamma i - \mu s \tag{1}$$

$$\Delta i = \beta si - \gamma i - \mu i \tag{2}$$

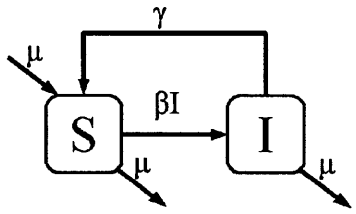


Figure 1 SI(S) Disease Model

Considering the mortality rate due to the disease, μ_i, the equations become:

$$\Delta s = \mu - \beta si + \gamma i - \mu s \tag{3}$$

$$\Delta i = \beta si - \gamma i - (\mu + \mu_i) i \tag{4}$$

This disease model undergoes a spatial adaption in STEM, since the populations at different areas are different. A new variable P_l is introduced, which stands for the total population in the area, and $P_l = S + I$. Moreover, it is assumed the transmission is more likely in a densely populated area, thus, the variability in β is added into the disease model. We use β_l instead of β, and let $\beta_l = \beta d_l / APD$. d_l is the population density of the area and APD is the average population density. Multiplying both sides of (3) and (4) using P_l, the equations becomes:

$$\Delta S = \mu P_l - \beta_l Si + \gamma I - \mu S \tag{5}$$

$$\Delta I = \beta_l Si - \gamma I - (\mu + \mu_i) I \tag{6}$$

Let $\beta^* = \beta_l / P_l = \beta\left(d_l / (APD * P_l)\right)$, they become

$$\Delta S = \mu P_l - \beta^* SI + \gamma I - \mu S \tag{7}$$

$$\Delta I = \beta^* SI - \gamma I - (\mu + \mu_i) I \tag{8}$$

SIR(S) Model [9]

SIR(S) model is usually used to model disease such that infected population can acquire immunity for a period before they become susceptible again. For pure SIR model, the

immunity can last for a life long time, which means once infected and recovered from the disease, the person won't be infected anymore. S and I carry the same meaning as SI(S) model, R refers to Recovered. The transition diagram for SIR(S) models is as follows. γ is the rate that an infectious person recovers from the disease and becomes immune. The immunity is lost at the rate of σ, for SIR model, this rate is 0.

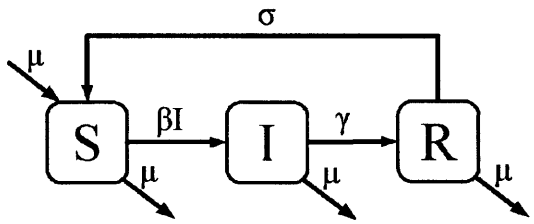


Figure 2 SIR(S) disease model

For normalized population, the relations of these variables are as follows.

$$\Delta s = \mu - \beta si + \sigma i - \mu s \tag{9}$$

$$\Delta i = \beta si - \gamma i - \mu i \tag{10}$$

$$\Delta \gamma = \gamma i - \sigma \gamma - \mu \gamma \tag{11}$$

After spatial adaption and change of variables, let $\beta^* = \beta_l / P_l = \beta\left(d_l / \left(APD * P_l\right)\right)$, the equations become:

$$\Delta S = \mu P_l - \beta^* SI + \sigma R - \mu S \tag{12}$$

$$\Delta I = \beta^* SI - \gamma I - \left(\mu + \mu_l\right) I \tag{13}$$

$$\Delta R = \gamma I - \sigma R - \mu R \tag{14}$$

SEIR Model [10]

In this model, S, I and R hold the same meanings as previously. E refers to exposed. For some disease, even if a person is exposed to a disease, it is not guaranteed that person will be infected. The rate that an exposed person becomes infectious is ε. The transition

diagram is as follows.

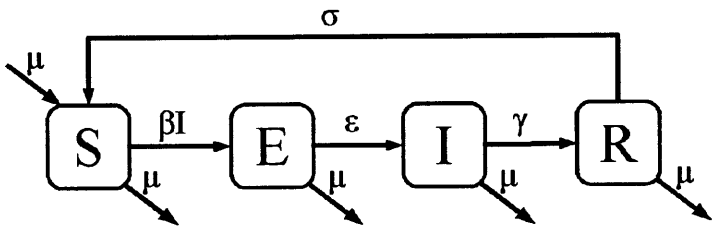


Figure 3 SEIR(S) Model

The relations formulas are

$$\Delta s = \mu - \beta si + \sigma\gamma - \mu s \tag{15}$$

$$\Delta e = \beta si - \varepsilon e - \mu e \tag{16}$$

$$\Delta i = \beta si - \gamma i - (\mu + \mu_i) i \tag{17}$$

$$\Delta \gamma = \gamma i - \sigma\gamma - \mu\gamma \tag{18}$$

After spatial adaption, they become

$$\Delta S = \mu P_i - \beta^* SI + \sigma R - \mu S \tag{19}$$

$$\Delta E = \beta^* SI - \varepsilon E - \mu E \tag{20}$$

$$\Delta I = \varepsilon E - \gamma I - (\mu + \mu_i) I \tag{21}$$

$$\Delta R = \gamma I - \sigma R - \mu R \tag{22}$$

Poisson disease model

A Poisson disease model has been proposed by Larson [3] which assumes the face to face contact is a Poisson process and associates the transmission rate with the parameter of this process. The model assumes a heterogeneous population and divides the whole population into several levels, according to their number of social contacts. The mathematics of the disease progression on a two-level population is defined as follows.

Define

n_H = the initial population of high-activity persons.

n_L = the initial population of low-activity persons.

λ_H = Poisson rate of social contacts per day of a high-activity person.

λ_L = Poisson rate of social contacts per day of a low-activity person.

$n_H + n_L$ = total population.

$n_H^I(t)$ = the number of infected high-activity person on day t.

$n_L^I(t)$ = the number of infected low-activity person on day t.

$n_H^S(t)$ = the number of susceptible high-activity person on day t.

$n_L^S(t)$ = the number of susceptible low-activity person on day t.

p = the probability that a susceptible person becomes infectious, given contact with an infectious person.

β = the next interaction of a person is with an infectious person.

$p_H^S(t)$ = the probability that a random susceptible high-activity person becomes affected on day t.

$p_L^S(t)$ = the probability that a random susceptible low-activity person becomes affected on day t.

We have

$$\beta = \left(n_H^I(t)\lambda_H + n_L^I(t)\lambda_L\right) / \left(n_H\lambda_H + n_L\lambda_L\right). \tag{23}$$

$$p_H{}^s(t) = 1 - e^{-\lambda_H \beta p}. \tag{24}$$

$$p_L{}^s(t) = 1 - e^{-\lambda_L \beta p}. \tag{25}$$

$$p_H{}^S(t) = 1 - e^{-\lambda_H \beta p}. \tag{26}$$

$$n_H^I(t+1) = n_H^S(t) p_H{}^S(t). \tag{27}$$

$$n_L^I(t+1) = n_L^S(t) p_L{}^S(t). \tag{28}$$

$$n_H^S(t+1) = n_H^S(t) - n_H^I(t+1). \tag{29}$$

$$n_L^S(t+1) = n_L^S(t) - n_L^I(t+1). \tag{30}$$

These equations can be applied iteratively on the population to get the updated number of infected and susceptible population. This model is in nature in line with the SI disease model introduced earlier on, except that it takes account into the heterogeneous nature of the population and the Poisson nature of human contacts.

The number of levels of social contacts can vary according to the characteristic of the population. A three level model has been used to measure the impacts of government posed and voluntary non-pharmaceutical interventions in [11]. In this paper, it indicates some important results that these forms interventions are effective in terms of mitigating the possible global outbreak.

2.3 Current epidemic modelling

After the disease model is designed, simulation is followed to test the characteristics of the disease and evaluate the effects of the control methods. Researchers usually design their own experiment procedures and apply their disease model into the experiments.

Since epidemic disease models are usually in the form of mathematical models, Monte Carlo method is often used in the simulations. First, the initial population and the parameters are determined. After initialization, the inputs go through repeated calculations according to the mathematical equations and reach a final result. Matlab® is usually used to conduct these simulations. It is vigorous in numerical computation and numerical results can depict the characteristics of disease progression.

Graphs and animations are useful aids to describe the progression of epidemic disease. For example, Jave applet for disease model simulation is developed by Shodor [12]. The user can input initial population, number of doctors, number of travel openings and the disease parameters, etc before starting the simulation. The user can also choose whether to have control methods in the simulation, such as the use of vaccinations and medications. Setting up a scenario with one person sick in the community initially, the progression of the disease is as follows. The susceptible, infectious, recovered and dead people are respectively green, red, blue and black. The number of holes on the walls of the separated regions is the number of travel openings.

The progression reaches a steady state in the last phase where all the people are either immune or susceptible.

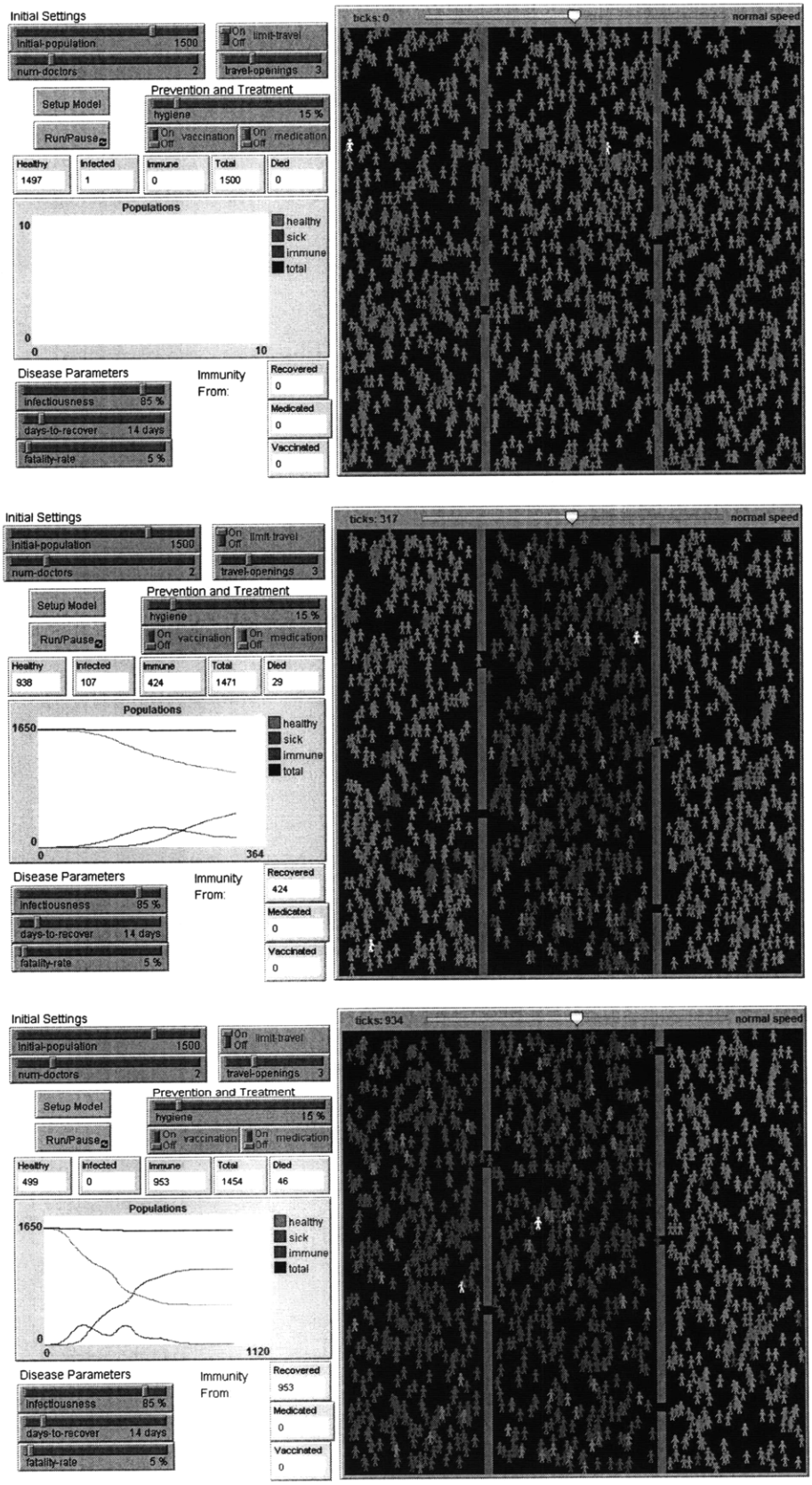


Figure 4 Shodor disease model Java applet

In 2004, Models of Infectious Disease Agent Study (MIDAS) was established by the National Institute of General Medical Sciences for research in novel computational infectious disease models [16]. They have developed various epidemic models for different modelling purposes. One example is a global epidemic model of H5N1 Influenza. The Java applet is as shown as follows. It has features of computing the disease model dynamics, R0 and choices of interventions to apply to the modelling. The information of infectious cases exporting through airlines is included in the model.

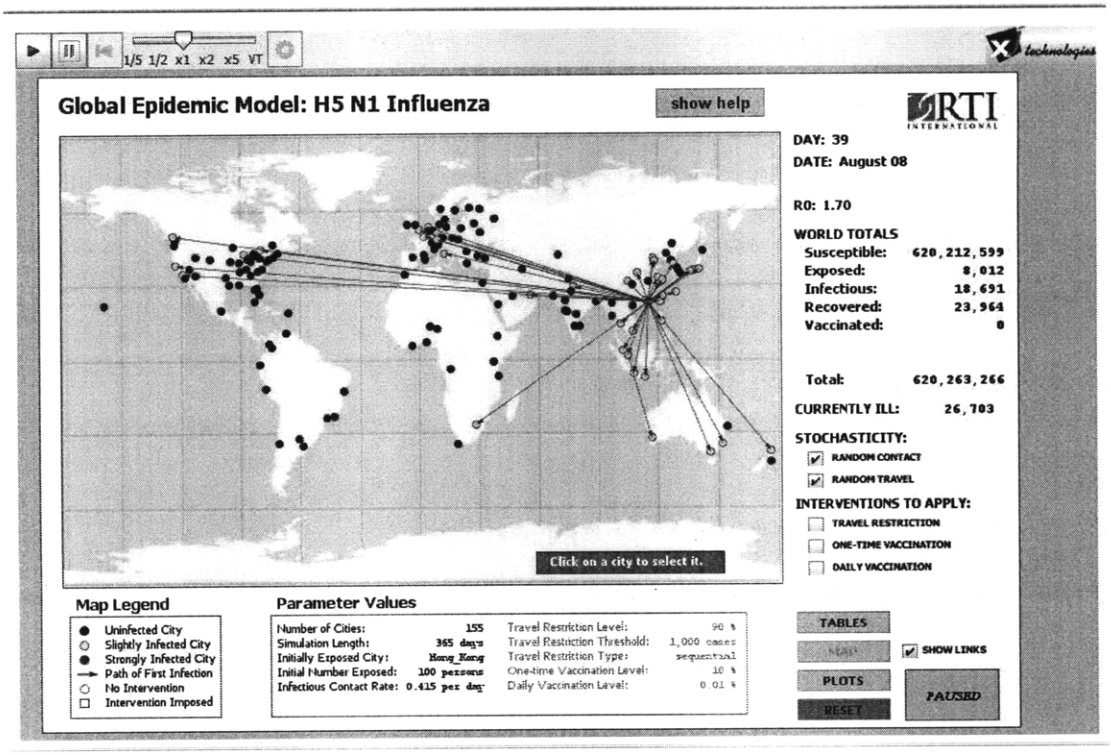


Figure 5 Global Epidemic Model by MIDAS

These simulation programs are helpful for the designers to proof their theoretical hypothesis. However, to be used as simulation tools which relates to the real world scenarios, they have some limitations.

First, they are usually based on a simple hypothesis of one or several community and a small population. The actual population information located on different areas of the world

is not incorporated in the simulation.

Second, the interactions of different communities are also simplified. For example, the interactions in the previous software are models as the number of holes opened on the walls. In real life cases, neighbour communities are connected through various transportation methods such as railways, highways and flights.

Lastly, these programs are usually predefined with their progression mathematics, namely the mathematical models, and there is not any flexibility of fitting other models into them. To do this, a separate version of the program has to be developed to solve the new problem.

In the last few years, the idea to improve the state-of-the-art disease modelling techniques and use visualization to better understand disease models led to the development of Spatiotemporal Epidemiological Modeller (STEM). STEM is an epidemic disease modelling software program developed by IBM .With data sets for countries defined using the ISO-3166 standard; it provides a platform to visualize the progression of an epidemic disease on an electronic map. It also has a gateway to Google Earth, which imposes the disease status and population information onto the map images from Google Earth. Another innovative feature of STEM is its flexibility and extensibility. Researchers can design different modelling scenarios through combining various disease parameters or implement their own disease models and plug them into STEM systems. All these models can be reused in other modelling scenarios or serve as basic models to be enhanced.

2.4 **Design specifications of STEM** [13]

Canonical Graph

The Canonical Graph is the basic element of STEM, which represents the information of geographic regions, the disease information at these locations and the relations between different regions. Similar to any other types of graphs, it is composed of edges, nodes and labels. Nodes usually represent geographic regions and edges usually represent the relationships of these geographic regions. The labels on nodes and edges contain information. The labels on nodes store the information of the title, the physical area, the size of population and the status of the disease in that region. If an edge represents the sharing of common borders between two regions, the label will contain information of the length of the common border. If an edge represents a road that connects two regions, the label on it might contain information of the type of road and how much the traffic is. In some other cases, the label might contain information of flight routes.

The information stored in the canonical graphs follows the ISO-3166 standard.[18] The countries and territories are denoted by the standard short alphabetic codes. There are three levels in the naming convention. The first level (Level 0) is country/territory; the second level (Level 1) is the province/state of a country; the last level (Level 2) is the counties/cities of the province/state, which are represented using Federal Information Processing Standard (FIPS) code in the US. An example for Santa Clara County in California is US-CA-06085. For those countries that the Level 2 codes are unknown, the system will generate a special code for that location till the correct code is determined.

STEM Components

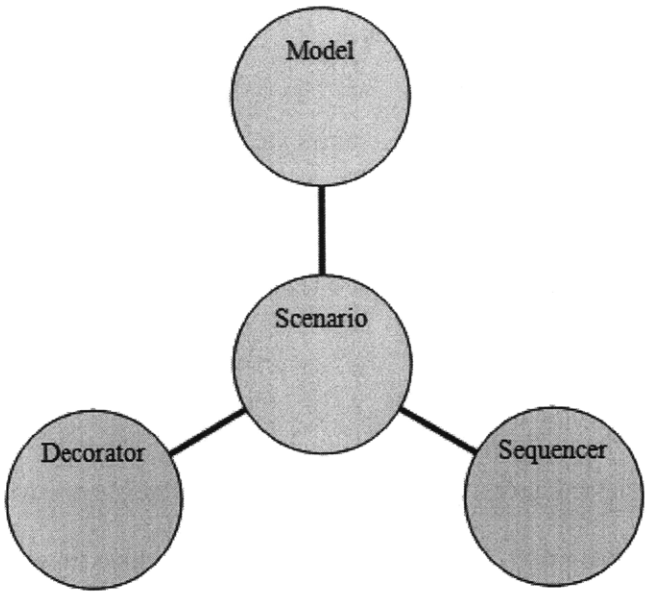


Figure 6 Components of a Scenario

There are all together eight building blocks of any simulation: Scenario, Sequencer, Decorator, Model, Graph, Modifier, Trigger and Predicates. Any STEM simulation is built from a Scenario. Scenario logically collects three components, namely Model, Decorator and Sequencer.

A Model is the component responsible for representing the contents of the canonical graph and for creating an instance of it when a Simulation is started from a Scenario. It combines the contents of Graphs and forms a tree. The tree is a hierarchical organization of the different contributions to the canonical graph.

A "Decorator" is the computational component in the STEM framework [13]. It initializes the canonical graph through setting the initial values of labels on the graphs and adding new labels. During any simulation, a decorator is responsible for computing the next graph status according to the disease model mathematics and the current simulation time. In

epidemiological simulation, a disease model is usually implemented as an decorator.

A "Sequencer" is a component that determines the sequence of time values which are used to compute the next stage of the canonical graph. It contains the information of the time interval, and duration of the simulation. In STEM, the time unit used is STEM time, which is equivalent to one or two days in real time.

A Graph contains the actual components, Nodes, Edges and Labels, which are eventually used to form canonical graph. Graphs are only fragments which may contain unresolved edges. The resulting canonical graph is the true mathematical graph.

A Modifier is a specification of how to change the values of one or more features of a Scenario. There are two types of modifiers, Range Modifier and Sequence Modifier. The former modifies values through assigning from a specified range of values. The latter does this through assigning them from a pre-specified ordered collection.

A Predicate is a Boolean expression that returns false or true depending on the results of any testable conditions. A Trigger is a special type of decorator which combines the results of a Predicate and a Decorator. It first evaluates the situation using a Predicate and then decides on whether to continue with the execution with the Decorator's description

Simulation

A Simulation is usually started from Scenario. First, a canonical graph is created through recursively descending the tree rooted by the Model. After the edges and nodes are created, other information are added through Decorators. Then, the simulation checks if it has

completed the sequence of cycles through the value of Sequencer. If it has not, the canonical graph will move to next stage, all the value of the labels are calculated according to the previous stage and the information stored in the Decorator. The simulation continues in this way until the end of the time specified by the Sequencer.

2.5 Eclipse Modelling Framework (EMF)

To generate Scenarios that are based on the textbook disease models introduced previously, SI(S), SIR(S) or SIER(S), the user can define their building blocks directly and construct them accordingly from STEM interface. However, if any new models or any modifications of the existing models are to be done, users have to design their own disease model plug-ins through the Eclipse Modelling Framework (EMF) and plug them into the application before constructing any Scenarios.

Eclipse Modelling Framework (EMF) [4] is a Java framework and code generation facility which builds tools and applications based on a structured model. Although EMF uses XML as its canonical form of a model definition, there are several ways to get your model into that form. You can choose to create the XML document directly or export it from a modelling tool. You can also generate it from annotate Java interfaces with model properties. STEM is bases on a Java application. Developers can utilize EMF modelling for code generation in pure Java development environment.

EMF modelling can increase the productivity of developer. After you specify an EMF model, the generator will create a structured series of Java classes correspondingly. You can modify these classes through implementing new methods and adding new variables. You can also regenerate the model after your modification. The changes you made to the code will preserve after the regenerating process. Moreover, the EMF generated model can easily be interpolated into other EMF generated applications. For example, the design framework of STEM is done through EMF, thus after a disease model is generated through EMF, it can be inserted into the application as a plug-in and it will be reflected in the new version of STEM run from this framework.

There are two fundamental frameworks in EMF. One is core framework and the other is EMF.edit. The first one is responsible for creating Java implementations of a model including both code generation and runtime support. The second one is responsible for generating supporting classes which enable viewing and editing of the model.

EMF generates the Java classes in core framework in the following way. First, the classes of the new model are created. This can be done through drawing a model using UML, writing the model in XML, or coding the model in annotated Java. After this is done, EMF will generate a Java interface and an implementation class for each class in the model. The interface will contain the get and set methods for each attribute. The details of these methods are described in the implementation classes. The get method returns an instance variable representing the attribute. The set method is an efficient way to set the value of the instance variable. Moreover, EMF also generates two more interfaces, a factory and a package. The factory is used to create instance of your model classes. The package contains all the static constants used by the methods in the model. You can also generate a class from a super class, which is inheritance. The generated interface can extend the super interface. Last but not least, you can add additional methods and not to worry about using these changes when you later want to regenerate the model.

After you implement the core framework, you can use EMF.edit to add user interface to your model. EMF.edit will generate a series of interfaces, such as the ItemProviderAdapterFactory classes, the ItemProvider classes, Editor, Plugin, plugin.xml and Icons. In the STEM case, the plugin.xml will link the disease model you created to the main body of STEM and include it when you run STEM in Eclipse modelling framework.

Most of the time, the generated classes and xml documents are subject to modifications before they can work properly together to create a plug-in.

EMF helps to create a Java implementation efficiently. It is also the tool to create user defined disease models in STEM. A detailed description on the creation of a disease plug-in will be included the latter chapters.

Chapter 3

Creating a Scenario and running simulations

3.1 Constructing a Scenario in STEM

As mentioned in Chapter 2, any simulation has to be started from a Scenario. Thus, creating a Scenario is the basic step before conducting any simulation.

From a disease modeler's point of view, a Scenario needs to contain the following elements: the disease model, the spatial restriction, the time periods of the disease progression and the initial triggering number and location of patient zeros. All these elements are reflected in the design of STEM.

The disease model, which is the mathematical model, can be created from the new disease icon. There are a few choices of disease models: SI, SIR and SEIR. Each of these disease models are of two types: deterministic and stochastic. User can input the parameters of the disease model that is to be used in this Scenario. These parameters include transmission rate, recovery rate, mortality rate and etc. There are also some other parameters to be inputted by the user, such as population type, population density etc. All these inputted parameters are editable later on even after the Scenario is created. There are two solver programs: Finite Difference and Runge Kutta Adaptive Step Size. Finite Difference is a faster program and can be used for a quick check to see if the simulation is as expected. However, when an accurate analysis on real data is required, the latter is recommended to be used.

The geographical data are reflected in the Graph element. The geographical data for around 270 countries and regions have been stored in STEM as graphs. There are choices of graphs depending on the accuracy of the simulation. You can choose from a 2-level graph which will reflect all the counties and cities in a country or just a 1-level graph which will divide the country into states/provinces only. To run the simulation in a region which might include several countries, you need to not only include the graphs of each of these countries, but also to include the data of the common borders of these countries. These information together define the region whether the disease progression is to be examined.

The time period of the simulation is defined by Sequencer. After clicking on the New Sequencer icon, a window in the form of a calendar will pop out. You can specify the start time and end time of the simulation. Alternatively, you can specify the start time and cycle period. The simulated Scenario will run through the specified period.

The last element is the triggering infectious population. This is described by Infector. You can input the information of the infector such as the number of people infected and the location of these people. These people are like seeds which started the epidemic disease progression process.

A newly developed feature is inoculators, which the user can define the proportion of the inoculated population in a specified region. These inoculated people have immunity of this disease and will not become infectious.

After all these elements are specified, they are accordingly organized in the following structure to create a Scenario.

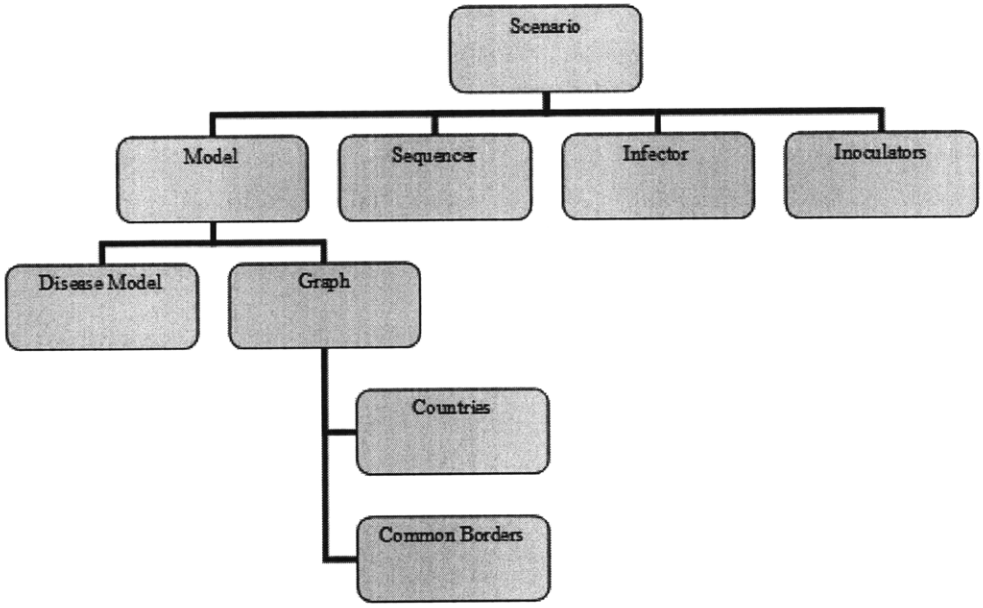


Figure 7 Structure of a created Scenario

After the Scenario is created, the user can modify the values of the elements through opening the element in the resource set in the upper right window, open it in the property editing window. The parameters are stored in the Dublin Core of the particular element. After making changes to the parameters in the window, you have to save these changes for them to be reflected in the model.

3.2 Running simulation

After a Scenario is created, the simulation can be started through right clicking on the Scenario icon and choose 'run'. A control panel will pop out as follows. The green arrow and yellow sign refer to run and pause the simulation respectively. The black button in the central refers to start over the simulation from beginning and the red square refers to stop and end the simulation. The map will keep updating itself until any pause button is pressed. Sometimes, a time discrete version of the graphs is needed by the user. The foot print button refers to 'step', which means the simulating is run in a step by step manner controlled by the user.

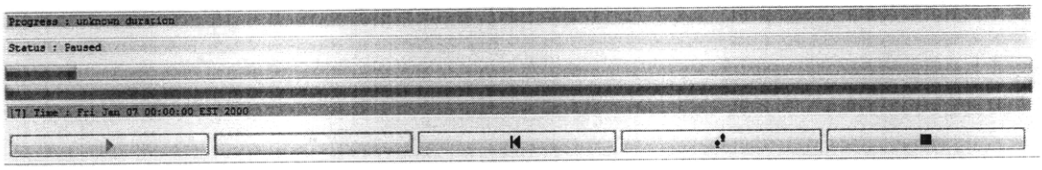


Figure 8 Control panel view

After a short initialization period, a map of the destination region will pop out. For example, a Spanish Flu case spreading all over Asia will show the full map of all the countries in Asia, their common borders and the edges of the counties in each of these countries. The map can be zoomed in to look at the details of the particular area. When you move the cursor to a particular point, it shows the information of that particular location, including the name, the area, the population and longitude and latitude.

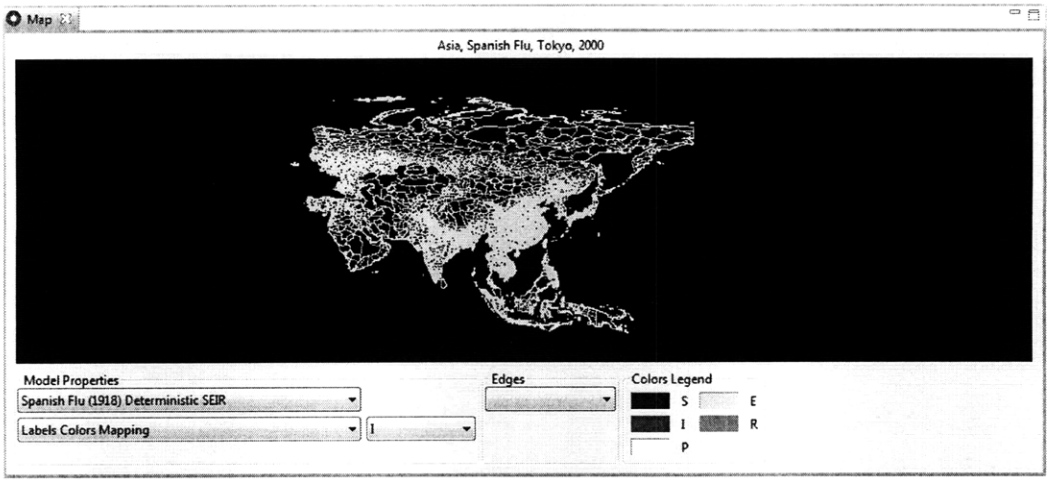


Figure 9 Initial Map view

The Infected regions are colored by red. The intensity of the color is proportional to the percentage of population that is infected. Thus, in the bright red region, the proportion of infected people is larger than the dark red area. After the simulation is running for sometime, most parts in Japan become green. There are no more red regions, which indicate that the disease dies out in the region and all the people are recovered from the disease. The map stabilizes at that point.

A time series analysis can be done at any point in the map. The proportions of infected, susceptible, recovered and exposed population are plotted on the graph with the x axis being the number of days. A generated time series graph is displayed in Figure 10. Time series analysis is extremely useful when the epidemic disease progression at a particular region is to be traced and examined. A phase space diagram can also be generated with any two of the parameters as x axis and y axis respectively. It is usually used to analyze the stability of the system.

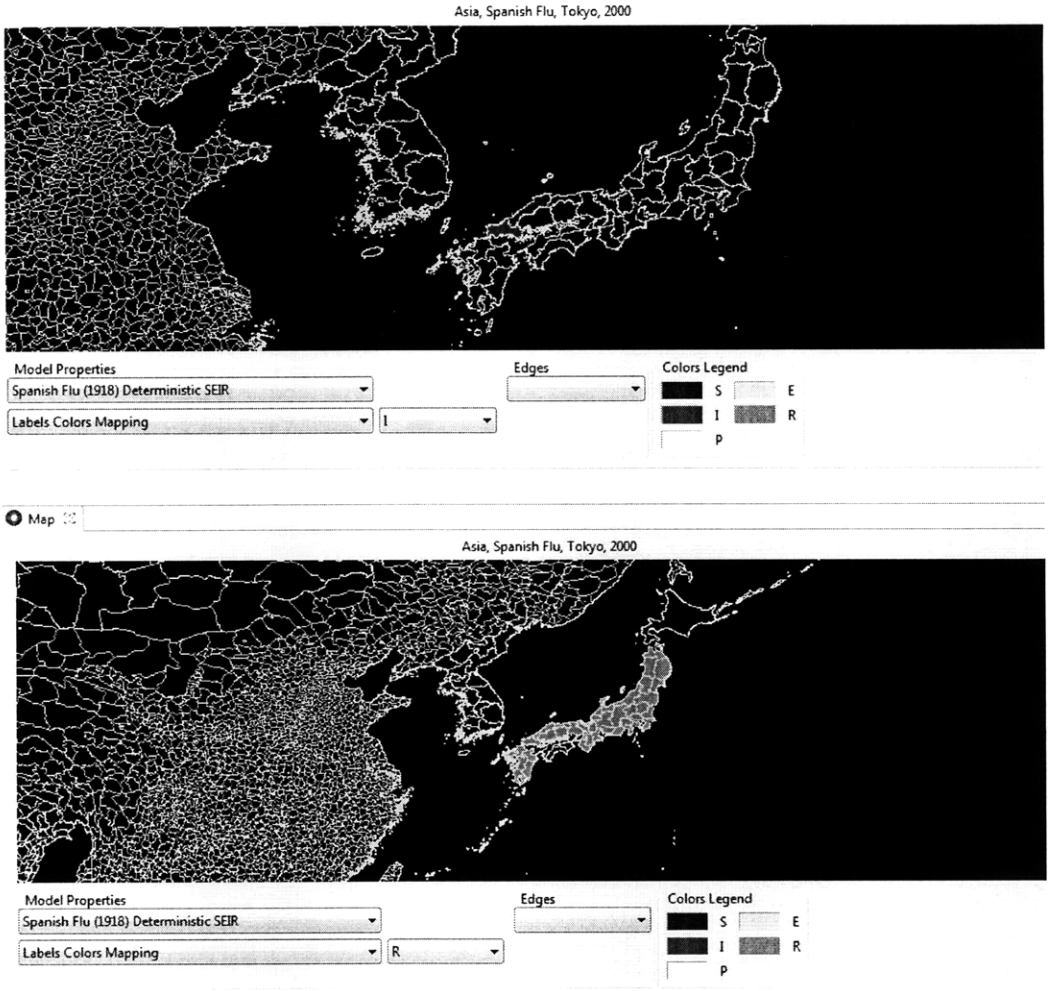


Figure 10 Infected region and recovered region

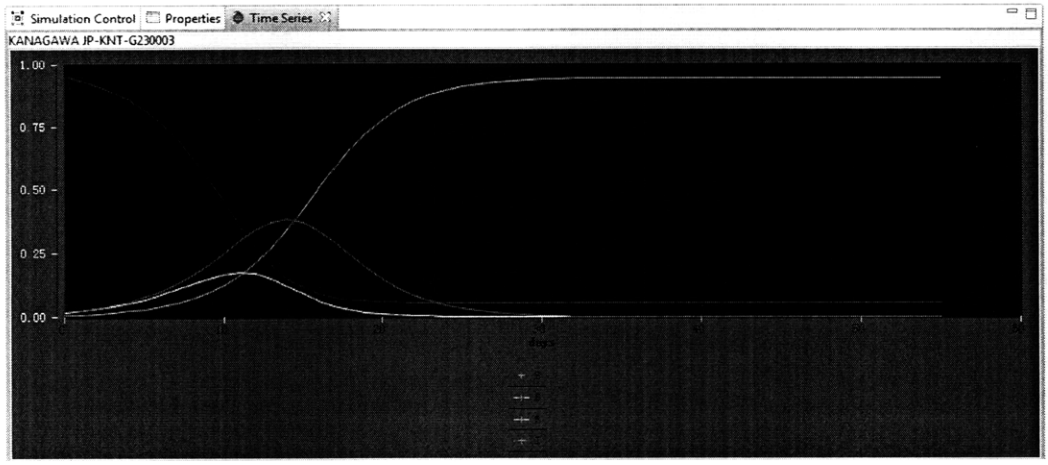


Figure 11 Time series analysis (with green being Recovered, blue being Susceptible ,Red being Infected, Yellow being exposed

3.3 Google Earth Interface

STEM also has a gateway to Google Earth® provided the software is installed. The current STEM builds haven't included this feature. The user has to plug-in the GE package in the software and run it from Eclipse to generate a new version of STEM.

In the newly generated version, you can choose from the menu, window -> GE view. After clicking it, Google Earth will be started automatically. However, unlike the embedded maps on which the disease progression is updated continuously with time, the views on Google Earth can only be generated discretely. In the middle of the simulation of a Scenario, you can click the GE icon on the left bottom. A control panel will pop up as follows.

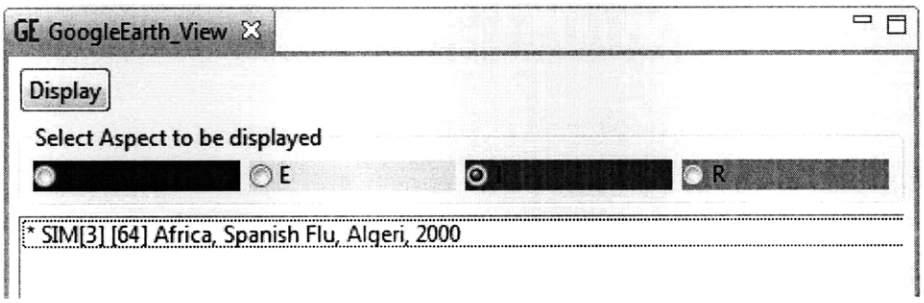
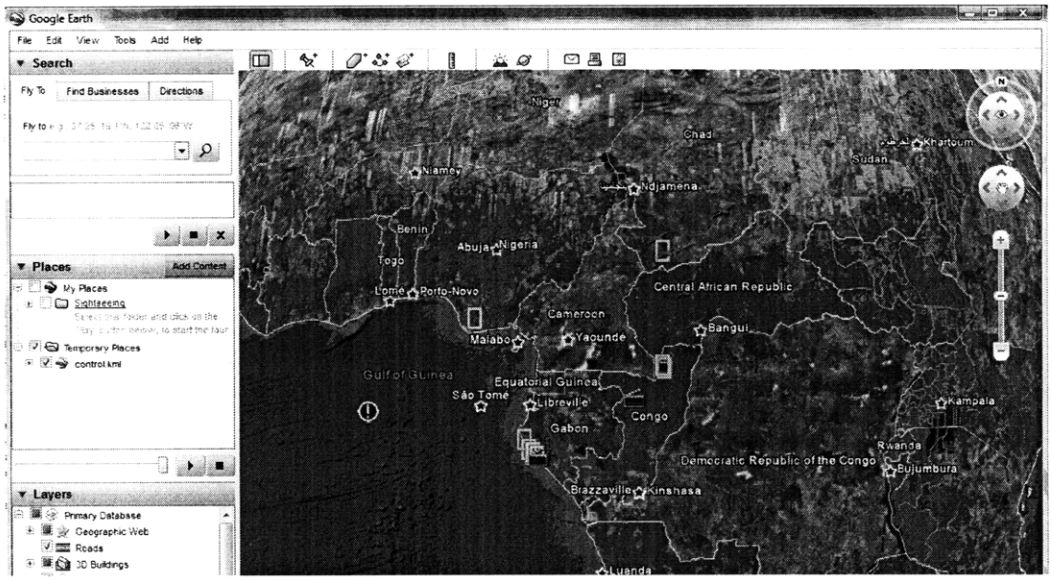
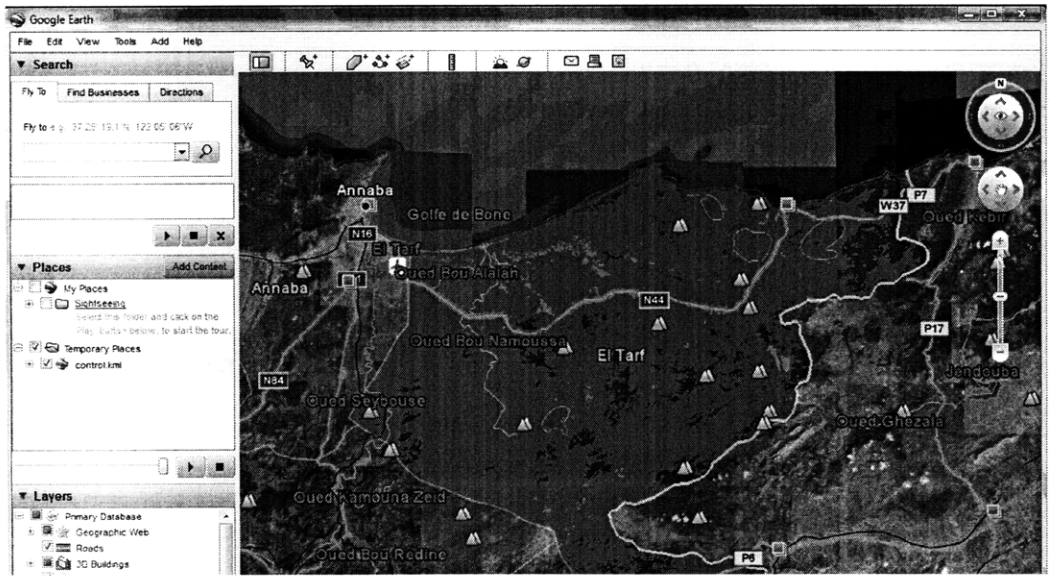


Figure 12 Google Earth Control Panel

After selecting the parameters you want to view on Google Earth®, and clicking display, a view on GE will be generated as follows. You can zoom in to view the details of the regions. Since GE maps provide more powerful functions than the implemented maps in STEM, this GE view helps the disease modeler and decision makers to conduct a detailed analysis of the disease progression in a certain region and some of the impacts of controls such as the number of hospitals near by, the amount of transportation everyday at the borders and between communities.


Google Earth
File Edit View Tools Add Help
Search
Fly To Find Businesses Directions
Places
Add Content
My Places
Sightseeing
Temporary Places
control.kml
Layers
Primary Database
Geographic Web
Roads
3D Buildings
Niger
Chad
Khartoum
Sudan
Niamey
Ndjamena
Benin
Abuja
Nigeria
Togo
Lomé
Porto-Novo
Central African Republic
Cameroon
Malabo
Yaounde
Bangui
Gulf of Guinea
Equatorial Guinea
São Tomé
Libreville
Gabon
Congo
Kampala
Rwanda
Democratic Republic of the Congo
Bujumbura
Brazzaville
Kinshasa
Luanda


Google Earth
File Edit View Tools Add Help
Search
Fly To Find Businesses Directions
Places
Add Content
My Places
Sightseeing
Temporary Places
control.kml
Layers
Primary Database
Geographic Web
Roads
3D Buildings
Annaba
Golfe de Bone
N16
El Tarf
Oued Bou Alalan
Annaba
N44
P7
W37
Oued Kebir
P17
Oued Bou Namoussa
El Tarf
N84
Oued Seybouse
Oued Ghezala
Oued Namouna Zeid
Oued Bou Redine
P9

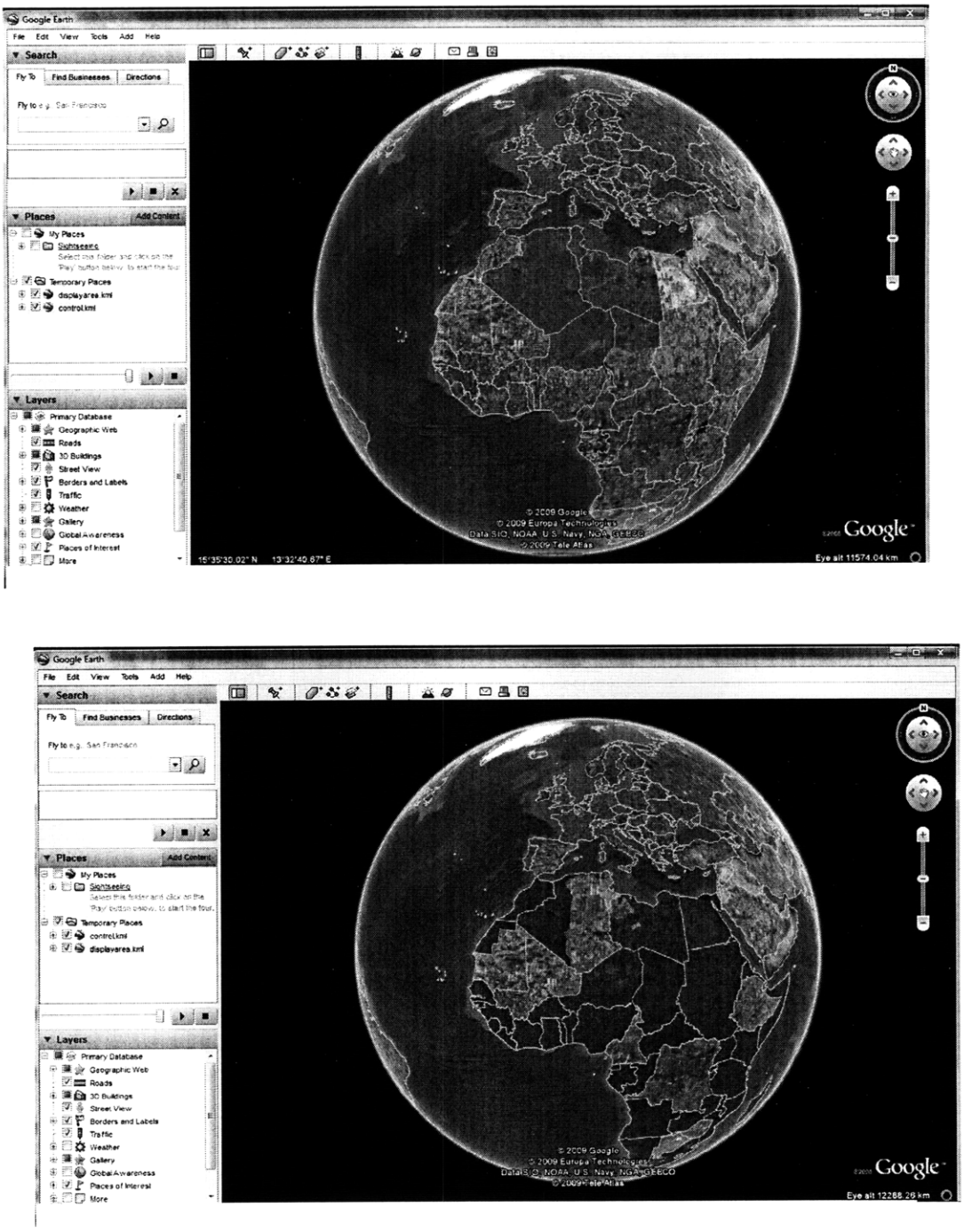


Figure 13 Google Earth view on the infected region and susceptible region

3.4 An example of employing an SI model into STEM

A hypothesis scenario is created and is simulated using STEM 4.0. Assuming an Influenza epidemic breaks out at Barnstable County, Massachusetts, USA, and 10 people are the first triggering patients. The epidemic progression follows an SI disease model with transmission rate of 2.0 and recovery rate of 1.0.

A Scenario called 'USInfluenza_case1' was created. The disease model 'USInfluenza' is created from a Deterministic SI Disease Model. A three-level graph is used for simulation. The location and number of people infected are included in the information of an infector called 'US_MA_infector'. These elements are organized in the project explorer window as follows.

After running the Scenario in the simulation view, the map keeps updating itself. It starts with a dark red area in the northeast corner of the US map. The red area keeps spreading quickly around that area. Since the transmission rate is 2.0, the epidemic disease progresses faster as time elapses. After one month, the adjacent states are infected and the disease keeps spreading to the left. Within 6 months, the influenza spreads all over the US.

As a testing case, the conditions used in this Scenario are simple and not realistic. However, this case still indicates that the disease cases increases in an exponential manner. Thus, controlling measures at initial stage of the breakout should be the most efficient.

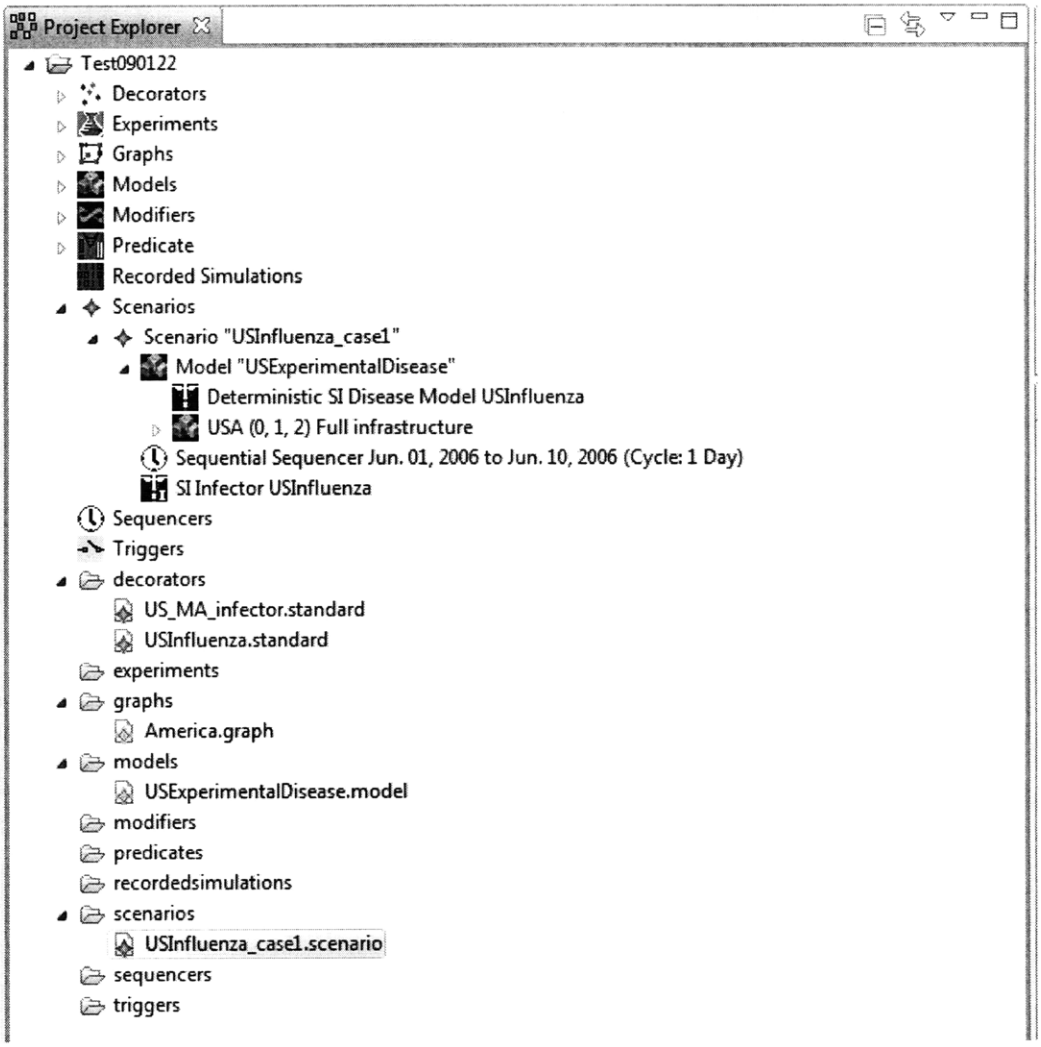
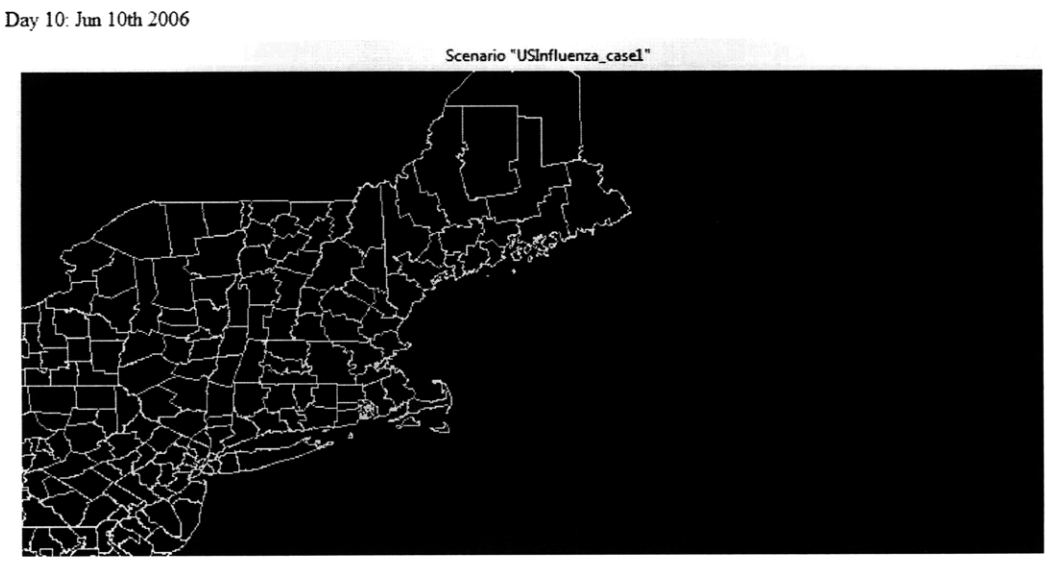
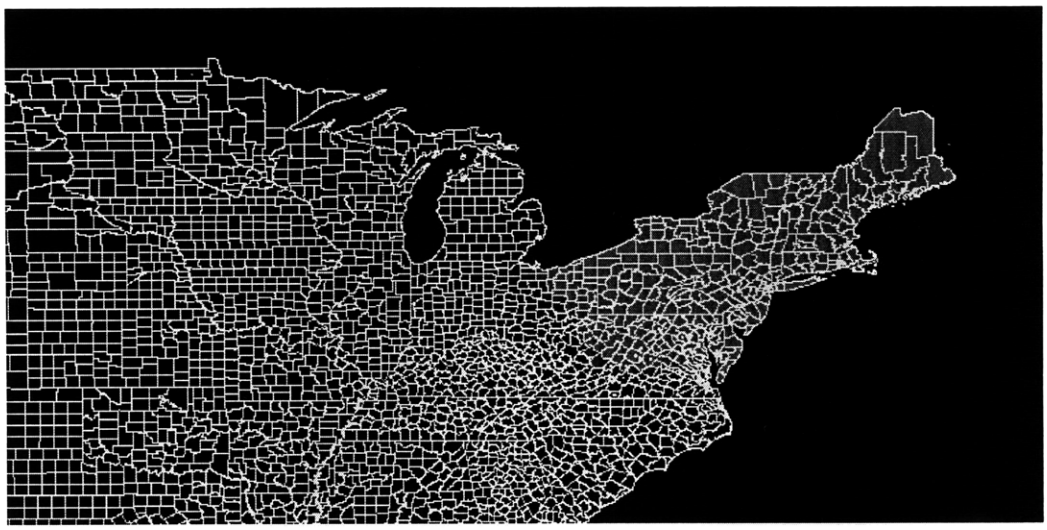


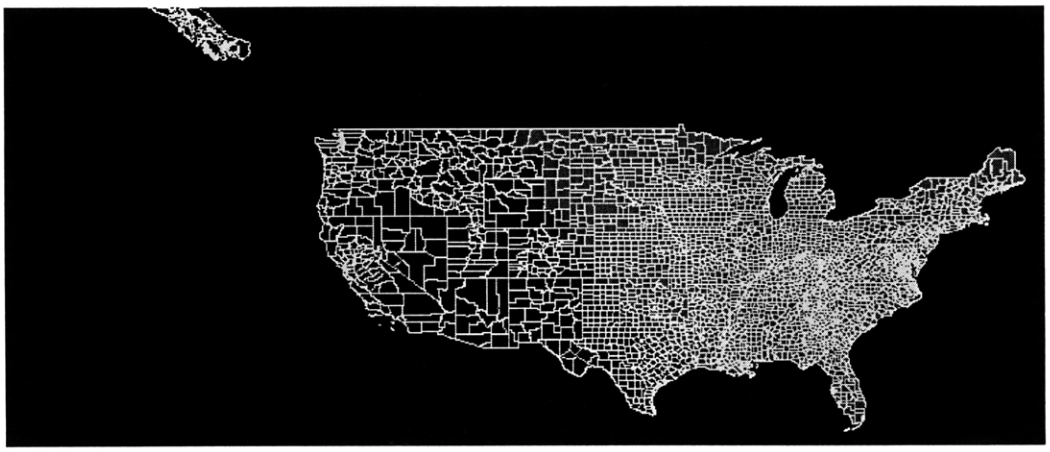
Figure 14 A view on the project explorer of Scenario 'USInfluenza_case1"



One month: Feb 1st, 2006



5 Months: May 11 2006



6 Months: Jun 10 2006

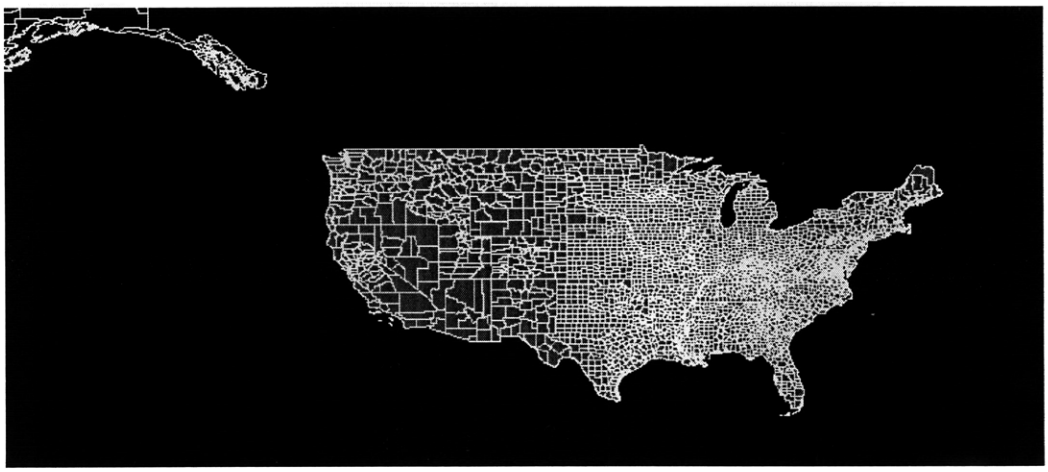


Figure 15 A hypothetical influenza break out in the US

Chapter 4

Employing a Poisson disease model of homogenous population

4.1 A Poisson disease model on a homogenous population

It is assumed in Larson's paper that the number of interactions in a day of a random person follows Poisson distribution with mean λ. In a homogenous population which assumes everyone has the same distribution for number of interactions every day, this characteristic defines the disease progression manner in the following way.

First, define the following input variables.

λ = Poisson number of contacts per day of a random person.

p = The probability of being infected given contact with an infectious person.

β = Probability that the next interaction is with an infected person.

We got to know that the probability that the next interaction is with an infected person β is the total number of contacts with an infected person divided by the total number of contacts with the entire population. Since the population is assumed to have the same contacts rate per day, β in this case is the same as the proportion of the infected population over the entire population.

Thus the probability that a random susceptible person becomes infected P_s can be derived accordingly. As a result, by the end of day k, the number of newly infected people on that day is the number of susceptible people times P_s. The total infected population is thus the

number of infected people before day k plus the newly infected people. The total susceptible population is the total population subtracted by the infected population.

Mathematically, they are represented as follows.

$P_s = 1 - e^{-\lambda \beta p}$.

$n_I(k)$ = number of infected people on day k = Infected population (I) * P_s.

I = total number of infected people $= \sum_{i=1}^{k} n_I(i)$.

S = total number of susceptible people = Total population - I.

From the equations above, we can notice a similarity between the Poisson model and the SI disease model mentioned earlier on. Thus, the idea of building the Poisson model on the basis of the SI disease model will be employed to realize the Poisson model.

4.2 The design hierarchy of disease models

To implement the new disease model, the developer has to download the source code of STEM and run it from Eclipse platform. The new models can be built on existing models. The hierarchy of the interfaces is displayed as follows.

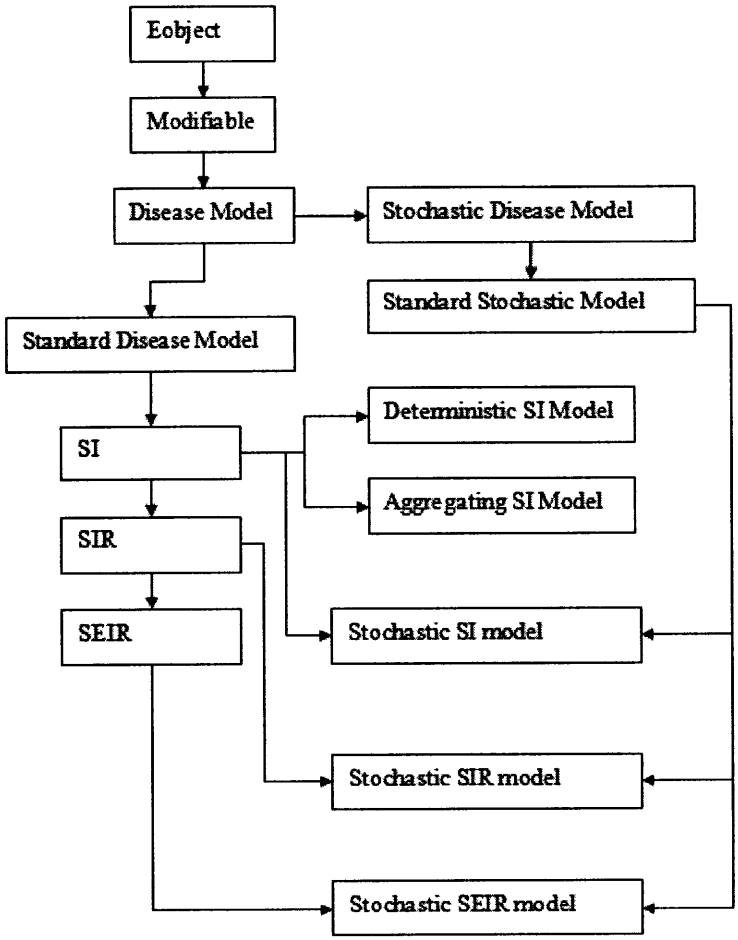


Figure 16 Hierarchy of disease model interfaces

Eobjects is the root of all modeled objects in the modeling framework. The disease model interface implements methods which read in parameters and construct the disease model. These parameters are user inputs such as disease name, time period, step size, frequency dependencies, etc. The standard disease model adds on some methods to get the total area, population and population density. The SI disease model has methods to read in

computation related data such as transmission rate and mortality rate, etc. The SIR and SEIR continue to build functions from SI disease model. Some of the interfaces inherit from two or more interfaces, such as the stochastic SI model, which inherits the features of both SI disease model and stochastic disease model.

It is crucial not only to understand the hierarchy, but also understand the functions of the parent interface, i.e. the methods already implemented. In this case, the Poisson disease model can be derived from SI disease model. Some changes have to be made to the disease dynamics and some extra parameters are to be added.

4.3 Creating the disease plug-in using EMF

To create the Poisson disease model using EMF, the first step is to run STEM as an application from Eclipse environment. The source code can be downloaded from Eclipse CVS repository [14]. The imported packages will show up in the Package Explorer window on the left hand side in Eclipse.

After that, the procedures of creating a disease model plug-in [15] are followed as follows.

Table 1 Procedures for creating a disease plug-in

Step	Procedure
1	Create a new EMF project
2	Create four packages
3	Define new disease model interface
4	Add Dependencies to MANIFEST.MF
5	Generate the EMF Model
6	Generate Model Code from EMF Model
7	Writing the Implementation Class
8	Edit autogenerated genmodel file
9	Generate the rest of the required code
10	Configuration of plugin.xml

In step 1, after making sure STEM can be started properly from the Eclipse environment, an EMF project called 'org.eclipse.ohf.stem.diseasemodels.poissondiseasemodel' is created. Right clicking on the project, four packages are created and named as follows:

org.eclipse.ohf.stem.diseasemodels.poissondiseasemodel;

org.eclipse.ohf.stem.diseasemodels.poissondiseasemodel.impl;

org.eclipse.ohf.stem.diseasemodels.poissondiseasemodel.provider;

org.eclipse.ohf.stem.diseasemodels.poissondiseasemodel.util;

In step 2, a new interface of poissondiseasemodel was declared as follows. It is important to include the declaration that the interface is an EMF Model, since it is responsible for code generation.

```
package org.eclipse.ohf.stem.diseasemodels.poissondiseasemodel;


import
org.eclipse.ohf.stem.diseasemodels.standard.DeterministicSIDiseaseModel;
/**
 * This interface is an EMF Model
 * @model
 *
 */
public interface PoissonDiseaseModel extends DeterministicSIDiseaseModel
{}
```

However, the classes extended will not be resolved. Thus the next step is to add dependencies to the MANIFEST.MF file. Clicking that file in the explorer, add the four dependencies as follows.

org.eclipse.core.runtime

org.eclipse.ohf.stem.core

org.eclipse.ohf.stem.definitions

org.eclipse.ohf.stem.diseasemodels.

Steps 5 and 6 generate the EMF Model and the model code to the structured position. Step 7 is the most important step, in which the classes and methods are written or modified to implement the functions of the new disease model. In Step 8, the paths of the edit code and editor code are modified in the genmodel file, which determines where these codes are to be placed. Step 9 generates the edit and editor codes. In the last step, plugin.xml is configured. This file is the connection of your disease model to the whole software design. After this, the disease model can be plug-in and the new version of STEM can run from Eclipse.

In PoissonDiseasemodelImpl, two new variables are introduced: the Poisson rate of contacts per day and the probability of infection, namely a person get infected given contact with an infectious person. The method in SI diseasemodel - computeDiseaseDeltas is modified to reflect the new mathematical logics on updating the infected and susceptible population at a certain location.

4.4 Showing your model in new disease model menu

Another package is required for showing your model in the new disease model selection, the org.eclipse.ohf.stem.ui.diseasemodels.poissondiseasemodel package.

Guidance on how to create a ui package in the project explorer is provided by James H. Kaufman from IBM Almaden Research Center and it listed in the Appendix.

Following the procedures, the two packages in the project explorer have the following structures.

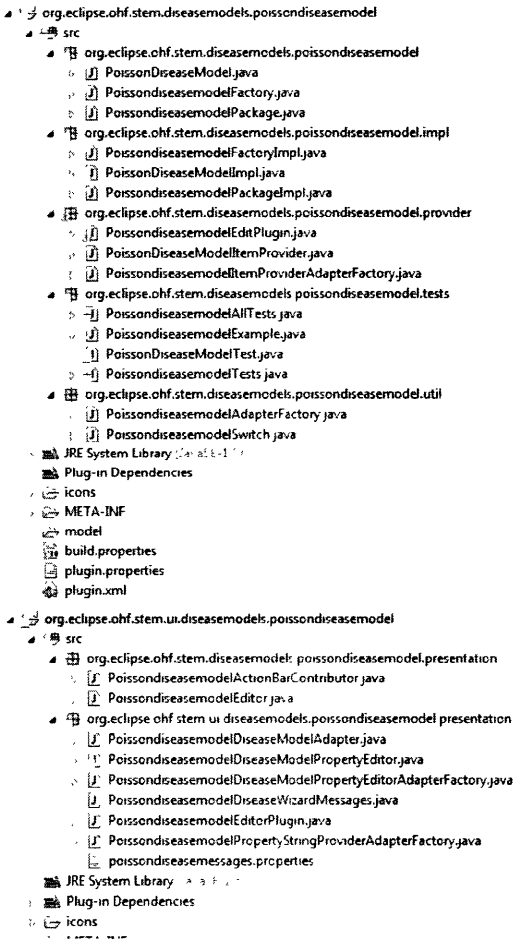


Figure 17 Poisson Disease Model Packages

Clicking run configuration, selecting both packages from the plug-ins and click run, a new version of STEM with the user defined will be started. Running the new disease wizard, the newly created Poisson Disease model shows up in the disease model menu. The user can create Scenarios as described in Chapter 3 and conduct simulations accordingly.

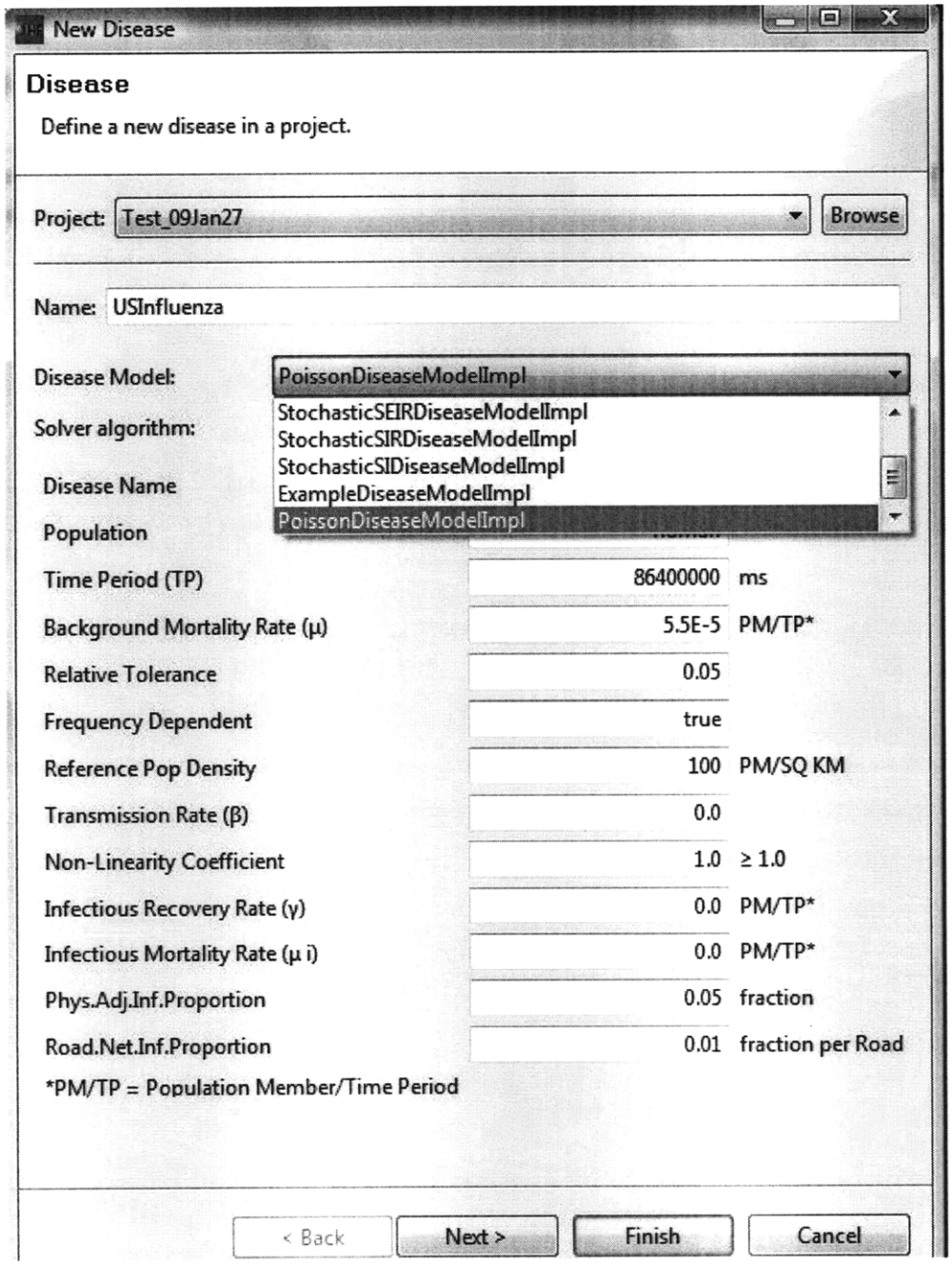


Figure 18 Poisson Disease Model show up in the menu

Chapter 5

Employing a heterogeneous disease model

5.1 A heterogeneous Poisson disease model

In Larson's paper, the Poisson disease model is constructed on a heterogeneous population which assumes that there are several different levels of activities of the total population. The number of daily contacts of a high activity person is a lot more than a low activity person. Introducing heterogeneity into the disease model reflects a more realistic scenario for epidemic disease progression. A three level disease model is used in [8] for analysis and some significance results on the application of NPIs are obtained.

The mathematics of the heterogeneous disease model are basically the same as the homogenous disease model, except that the probability that the next interaction is with an infected person β is more complex.

Assume the total population is divided into N levels according to their average number of daily contacts.

Define

λ^i = Poisson number of contacts per day of a random person in level i, $1 \le i \le N$.

n^i = The number of total population in level i, $1 \le i \le N$.

n_I^i = The number of infectious population in level i, $1 \le i \le N$.

n_S^i = The number of susceptible population in level i, $1 \le i \le N$.

p = The probability of being infected given contact with an infectious person.

β = Probability that the next interaction is with an infected person.

We have

$$\beta = \sum_{1}^{N} \lambda' n_I^i / \sum_{1}^{N} \lambda' n^i .$$

$$P_s^i = 1 - e^{-\lambda' \beta p} .$$

$n_I(k)$ = number of infected people on day k = $\sum_{1}^{N} n_I^i p_s^i$.

I = total number of infected people = $\sum_{i=1}^{k} n_I(i)$.

S = total number of susceptible people = Total population - I.

All the disease models implemented in STEM are based on a homogenous population, which assumes each individual in the population shares the same characteristics.

5.2 An algorithm to construct a heterogeneous population model

As mentioned earlier on, the information of the disease and population are stored on the labels in each node on the canonical graph. To reflect the heterogeneity of the population, a new disease label can be created which divides the total population into several levels. The Label hierarchy is designed in the following way.

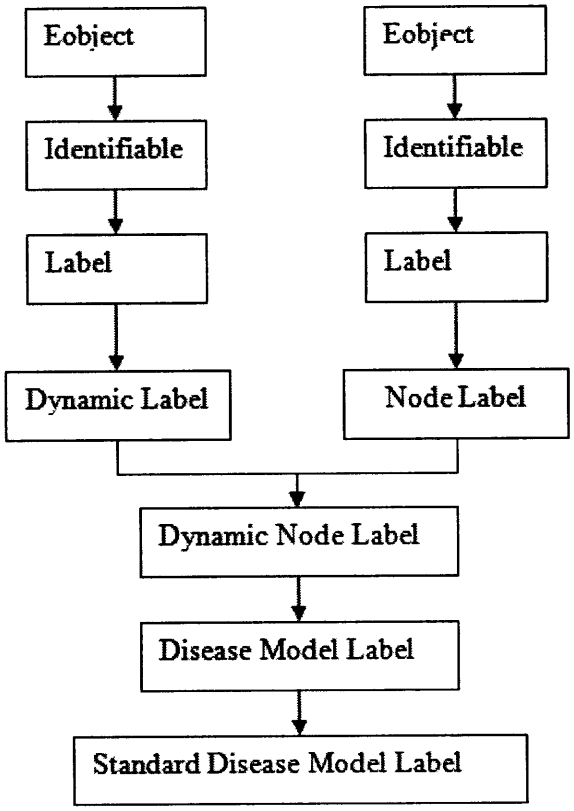


Figure 19 Hierarchy of disease label

In Disease Model Label, there is a method getPopulation, which returns an object PopulationLabel. From org.eclipse.ohf.stem.definitions.labels, it creates a population label and defines the type, name and populated Area. Thus, an algorithm can be proposed as following to include the feature of the heterogeneity.

First, a HeterPopLabel can be created as an extension class of the Population Label, which defines extra characteristics of the population, such as the level of activities, the proportions of each level in the population.

Next, a new disease label can be created as an extension class of the Dynamic node label. We can name it HeterDiseaseLabel, which implements the features of getting the detailed population information through HeterPopLabel.

After that, the steps from Chapter 4 can be followed to create two packages for the new disease model plug-in. In the new disease implementation, the method computeDiseaseDeltas will include the mathematics that applies to the heterogeneous population.

However, the proposed algorithm will change the basics of STEM design. Thus, the implementation of this algorithm would be tedious and involves a great effort in coding and debugging.

Chapter 6

Discussion

6.1 Evaluation of current build of STEM

The current released version STEM 0.5.0 is ready for download on STEM website. It is a stable build which has all the features of building a scenario from one of the implemented disease model. IBM's target is to release the 0.6.0 version in August 2009, which will include the functions of analysis perspective such as the time series analysis and phase space analysis. So far, the implementations of these functions are not stable.

The idea and realization of composing scenarios through several elements is bright and this function will help researchers in various ways. For researchers who are working on modeling using standard disease models, it undoubtedly saves effort in building the epidemic modeling simulation tools from scratch.

However, the flexibility of employing user defined disease models is so far still limited. On one hand, the documentations and guidance on creating user defined disease model are incomplete. On the other hand, the realization of extending the disease model is tedious and requires high demand of programming proficiency. For modelers who are not familiar with EMF modeling, the task is challenging.

6.2 Areas of application

STEM can be used by researchers and scientists when the effect of epidemic disease progression in a realistic geographic and populated region is to be evaluated and demonstrated. It can also be used to model large epidemics which have a global effect. Healthy care authorities may also find it useful, since the disease progression depicted by the dynamic maps introduced in STEM would help them to determine the severity of the epidemic disease and to evaluate the effectiveness of various control methods.

Although as an extensible platform, STEM still needs further improvements. Modeling using EMF saves a lot of effort for developers from direct building each modules of the software. Disease modelers may find it easier than to build their own simulation tools using their own disease model.

Chapter 7

Future work

7.1 Implementation of the algorithm

The current implementation of the homogenous model is partial. Due to lack of documentation, some issues over the creation of new parameter variables still need to be solved. Moreover, only the algorithm to implement the heterogeneous population model is proposed. The algorithm needs to be implemented into the software and verified. Future wok can be done according to the steps listed in Table 2 and Table 3. There might be some unexpected errors that occur in the process of EMF model generation and the descriptions might not include them. It is expected the programmer has a high proficiency in Java and has some experience in working with Eclipse and EMF.

Table 3. The steps for implementing a heterogeneous model.

Step	Description
1	Follow the algorithm in 5.2 and create a HeterPopLable as an extension class of the Population Label.
2	Create a HeterDiseaseLabel which implements the features of getting detailed population information through HeterPopLabel.
3	Create the Poisson disease model package according to Table 1 and [15].
4	In Step 7, rewirte the method computeDiseaseDeltas to include the mathematics of the Poisson Disease model, let the return type be the HeterDiseaseLabel.
5	Follow the Steps to create a ui package in Appendix A.

6	Declare the parameters in PoissonDiseaseModel.java in the following format.
	`/**`
	` * @return the default probability of infection.`
	` * @model default="0.1"`
	` */`
	`double getProbabilityOfInfection();`
	Write the parameters in PoissonDiseaseMessages.property in the ui package.

7	Correct any errors caused after the model generation process.
8	Run STEM and test the newly created disease model.

7.2 Potential improvements

First, as mentioned earlier, better documentations and guidelines need to be prepared for users to help the understanding and utilizing the functions.

Moreover, some new functions need to be built for users to create practical scenarios. For example, in 2009 H1N1 flu spread, most of the new cases are created through air transportation. However, this important feature is not included in STEM. The disease progression in STEM is mostly assumed to be proceeded across borders through roads.

Table 3. The steps for implementing a heterogeneous model.

Step	Description
1	Follow the algorithm in 5.2 and create a HeterPopLable as an extension class of the Population Label.
2	Create a HeterDiseaseLabel which implements the features of getting detailed population information through HeterPopLabel.
3	Create the Poisson disease model package according to Table 1 and [15].
4	In Step 7, rewirte the method computeDiseaseDeltas to include the mathematics of the Poisson Disease model, let the return type be the HeterDiseaseLabel.
5	Follow the Steps to create a ui package in Appendix A.

6	Declare the parameters in PoissonDiseaseModel.java in the following format.
	`/**`
	` * @return the default probability of infection.`
	` * @model default="0.1"`
	` */`
	`double getProbabilityOfInfection();`
	Write the parameters in PoissonDiseaseMessages.property in the ui package.

7	Correct any errors caused after the model generation process.
8	Run STEM and test the newly created disease model.

7.2 Potential improvements

First, as mentioned earlier, better documentations and guidelines need to be prepared for users to help the understanding and utilizing the functions.

Moreover, some new functions need to be built for users to create practical scenarios. For example, in 2009 H1N1 flu spread, most of the new cases are created through air transportation. However, this important feature is not included in STEM. The disease progression in STEM is mostly assumed to be proceeded across borders through roads.

To conclude, STEM has innovative new features which could make it an extensible modeling and simulation tool. However, it is hard to cater for special modeling needs which would still rely on specifically designed simulation tools.

References

1] Ford, Alexander Daniel, Kaufman, James H., Iris, Eiron, 2006. An extensible spatial and temporal epidemiological modelling system. International 516 Journal of Health Geographics 5, 4. Available from <http://www.ijhealthgeographics.com/content/5/1/4>.

[2] Isham, V, 2005. Stochastic models for epidemics: Current issues and developments. A. C. Davison, Y. Dodge, N. Wermuth, eds. Celebrating Statistics. Oxford University Press, 27–54. Research report 263, 2004. Available from <http://www.ucl.ac.uk/stats>.

[3] Larson, R.C., 2007. Simple models of influenza progression within a heterogeneous population. Operations Research 55 (3), 399–412.

[4] Budinsky Frank, Dave Steinberg, Ed Merks, Ray Ellersick, Timothy J. Grose, 2003. Eclipse Modeling Framework. Addison-Wesley Professional.

[5] Martin A, Krauss C, 2009.Pork Industry Fights Concerns over Swine Flu.New York Times. Available from <http://www.nytimes.com/2009/04/29/business/economy/29trade.html>.

[6] Taubenberger JK, Morens DM, 2006. 1918 influenza: the mother of all pandemics. Emerg Infect Dis Vol. 12, No. 1. Available from <http://www.cdc.gov/ncidod/EID/vol12no01/05-0979.htm>.

[7] Center for disease control and prevention, 2008. Avian Influenza: Current H5N1 Situation. Available from <http://www.cdc.gov/flu/avian/outbreaks/current.htm#humans>.

[8] The Eclipse Foundation, 2007. SI Disease Model Mathematics. Available from <http://ftp.daum.net/eclipse/technology/ohf/stem/help/latest/epidemiologicalmodeling/simath.html>.

[9] The Eclipse Foundation, 2007. SIR Disease Model Mathematics. Available from <http://ftp.daum.net/eclipse/technology/ohf/stem/help/latest/epidemiologicalmodeling/sirmath.html>.

[10] The Eclipse Foundation, 2007. SEIR Disease Model Mathematics. Available from <http://ftp.daum.net/eclipse/technology/ohf/stem/help/latest/epidemiologicalmodeling/seirmath.html>.

[11] Nigmatulina, K.R., Larson, R.C., 2008. Living with influenza: Impacts of government imposed and voluntarily selected interventions, European Journal of Operational Research, doi:10.1016/j.ejor.2008.02.016.

[12] The Shodor Education Foundation, Inc. , 2008. A Disease Model. Avalailbe from <http://www.shodor.org/featured/DiseaseModel/model/>

[13] The Eclipse Foundation.,2009. STEM Design Document. Available from <http://wiki.eclipse.org/STEM_Design_Document>

[14] The Eclipse Foundation, 2009. STEM Source Code. Available from <http://wiki.eclipse.org/STEM_Source_Code>

[15] The Eclipse Foundation, 2009. Creating a new Disease Model Plug-in. Available from <http://wiki.eclipse.org/Creating_a_new_Disease_Model_Plug-in>

[16] Models of Infectious Disease Agent Study, 2007. MIDAS Global Epidemic Model. Available from <https://www.epimodels.org/midas>

[17] Gross, D, 2009. Regular flu has killed thousands since January. CNN news. Available from
<http://www.cnn.com/2009/HEALTH/04/28/regular.flu/index.html>

[18] International Organization for Standardization, 2009. English country names and code elements. Avalable from
<http://www.iso.org/iso/english_country_names_and_code_elements>

[19] The Eclipse Foundation, 2009.The Spatiotemporal Epidemiological Modeler (STEM) Project. Available from <http://www.eclipse.org/stem/>

[20] Google Inc. ,2009. Google Earth. Available from <http://earth.google.com/>

Appendix

A1 How to create a ui package

1) Make sure the constructor in the class "YourdiseasemodelImpl()" is

PUBLIC. By default this is constructed as protected so you need to change

it.

Add the javadoc

/**

* <!-- begin-user-doc -->

* <!-- end-user-doc -->

* @generated NOT

*/

above the constructor so it does not get changed back if/when you

regenerate.

2) Open yourdisease.genmodel with an xml editor

3) Expand the root node

4) Change the editor directory to

/org.eclipse.ohf.stem.ui.diseasemodels.yourdisease/src

5) Click on the root genmodel:GenModel node, right click, and

add attribute

editorPluginClass="org.eclipse.ohf.stem.ui.diseasemodels.yourdisease.presentation.Yourdi

seaseEditorPlugin"

6) Click on the root genmodel:GenModel node, right click, and
add attribute
editPluginID="org.eclipse.ohf.stem.diseasemodels.yourdisease"

7) Click on the root genmodel:GenModel node, right click, and
add attribute
editorPluginID="org.eclipse.ohf.stem.ui.diseasemodels.yourdisease.editor"
click on the top node, right click, and one at a time rengenerate
everything.

Please recheck Make sure the constructor in the class
"YourdiseasemodelImpl()" is PUBLIC. By default this is constructed as
protected so you need to change it.

8) Delete
org.eclipse.ohf.stem.ui.diseasemodels.yourdiseasemodel.presentation.YourdiseaseWizard.j
ava

9) In the package
org.eclipse.ohf.stem.ui.diseasemodels.yourpoissondiseasemodel.presentation.presentation.
YourdiseaseEditorPluging
will have been created.
You need to create 5 classes in

org.eclipse.ohf.stem.ui.diseasemodels.yourpoissondiseasemodel.presentation

YourdiseaseDiseaseModelAdapter.java

YourdiseaseDiseaseModelPropertyEditor.java

YourdiseaseDiseaseModelPropertyEditorAdapterFactory.java

YourdiseaseDiseaseWizardMessages.java

YourdiseasePropertyStringProviderAdapterFactory.java

The easiest way to do this is to copy those files from the "Example"

disease model and rename

Note that you must to a CASE SENSITIVE replacement.

'example' -> "yourdieasename"

'Example' -> "Yourdieasename"

'EXAMPLE' -> "YOURDISEASENAME"

10) You must also create a file

yourdiseasediseasemessages.properties do define constant strings for

display in the property editor.

Please see the one in the example project.

11) Make changes to plugin.xml

open the plugin.xml file for

org.eclipse.ohf.stem.ui.diseasemodels.yourpoissondiseasemodel.presentation

and go to dependencies. You need these four only.

org.eclipse.ohf.stem.diseasemodels.yourdisease

org.eclipse.emf.ecore.edit

org.eclipse.ohf.stem.ui

org.eclipse.ohf.stem.ui.diseasemodels (ADD THIS ONE)

go to the Extensions tab and add

org.eclipse.ui.startup

under Extension Element Details for startup add the class

org.eclipse.ohf.stem.ui.diseasemodels.yourdisease.presentation.YourdiseasePropertyString

ProviderAdapterFactory

go to the Overview Tab. Change the activator class to:

org.eclipse.ohf.stem.ui.diseasemodels.yourdisease.presentation.YourdiseaseEditorPlugin$I

mplementation

Change the ID: to

org.eclipse.ohf.stem.ui.diseasemodels.yourdisease

Make sure your execution envitronment required java1.5

this should also be true for your disaease model plugin.xml file

====

12) In the class

org.eclipse.ohf.stem.ui.diseasemodels.yourpoissondiseasemodel.presentation.YourPoissond

iseasemodelPropertyEditor

in the METHOD

populate()

you must custom code the properties you wish to show up in the creation

wizard and diseasemodel property editor.

13) Finally,

open run configurations in the eclipse environment and make sure both your new disease model plugin AND

org.eclipse.ohf.stem.ui.diseasemodels.yourdisease are both checked. Then click "add required"