We present an integrated interactive modeling environment to support public health epidemiology. The environment combines a high resolution individual-based model with a user-friendly interface that allows analysts to explore various counter factual scenarios. The environment is based on a loosely coupled service oriented architecture. This allows a user to access the models and the analytics backend remotely from a desktop or a mobile device. As the modeling tools get more sophisticated, it is becoming increasingly harder for non-computational scientists to effectively use the system. Thus an important design consideration is to improve user productivity and ease of use. This is achieved by designing intuitive and user-friendly interfaces that allow users to design and analyze a computational experiment and steer the experiment based on the state of the system.

A key feature of the system that supports this design goal is its ability to start, stop, pause and roll-back the disease propagation and intervention application process interactively. An analyst can access the state of the system at any point and formulate dynamic interventions based on the additional information obtained through state assessment. In addition, the environment provides automated services for experiment set-up and management, thus reducing the overall time of conducting end-to-end experimental studies.

We illustrate the applicability of the system by describing computational experiments based on realistic pandemic planning scenarios. The experiments are designed to demonstrate the system capability and enhanced user productivity.

Categories and Subject Descriptors: I.6 [SIMULATION AND MODELING]: Applications; D.2 [SOFTWARE ENGINEERING]: Software Architectures, Data abstraction

General Terms: Design, Experimentation, Performance

Additional Key Words and Phrases: Computational epidemiology, Network-based epidemiological modeling, Computational steering, User productivity, Interactive computations, Usability, Service oriented architectures

1. INTRODUCTION

Epidemiology aims to study the spatio-temporal patterns of health in a population and the factors that contribute to these patterns. Computational Epidemiology is the development and use of computer models to generate appropriate spatio-temporal patterns as well as methods for controlling these patterns. Computational models may be descriptive, e.g. static estimates of correlations within large databases, or generative, e.g. computing the spread of a disease via person-to-person interactions through a large population. The infectious disease may represent an actual disease, or it may represent a more general reaction-diffusion process, such as the diffusion of innovation. The populations of interest depend on the disease, including humans, animals, plants, and computers. Similarly, the interactions that must be represented depend on the disease and the populations, including physical proximity for aerosol-borne disease, sexual contact for sexually transmitted diseases, and insect feeding patterns for mosquito-borne diseases.

Computational epidemiology allows analysts and epidemiologists to undertake counterfactual insilico experiments as well as study the efficacy of various intervention strategies. Potential interventions for controlling infectious disease include pharmaceuticals interventions, social distancing designed to reduce interactions between individuals, and eradication of vectors. Efficient use of these interventions requires targeting critical subpopulations that inhibit disease eradication.
spread. Computational models can be used to identify such critical subpopulations. The models can also be used to assess the feasibility and effectiveness of proposed interventions.

Useful computational environments that support epidemiologists need to satisfy important requirements, including: (i) model validity, (ii) computational efficiency, (iii) ability to represent a wide variety of diseases and interventions and (iv) ease of use. Some of the requirements are often in conflict and thus are challenging.

1.1. Contributions

In this paper, we describe the architecture and a prototype implementation of DISimS (Distributed Interactive Simulation System). DISimS is a flexible epidemiological modeling environment. DISimS combines high resolution individual-based epidemic and intervention modeling environment with a web-based user-friendly analytics environment. DISimS can be used by policy makers and epidemiologists for undertaking a broad range of counterfactual computer experiments. Additionally, the modeling environment can also be used for training analysts in the use of such models.

DISimS is an interactive modeling environment and interactivity is one of its key technical strengths. DISimS allows an analyst to start, stop, pause, resume and roll back previously applied intervention strategies and disease propagation process. Users can ask complicated spatio-temporal queries that support situation assessment. DISimS aids policy makers interested in developing dynamic health policies – policies that can adapt to new data that becomes available via surveillance. This is an important issue in epidemiology. See a recent paper by Yaesoubi and Cohen [Yaesoubi and Cohen 2011] for additional discussion. Developing such interactive simulations and computational steering environments, especially for parallel simulations is a well known challenging problem. DISimS achieves this by exploiting the problem specific semantics that allow one to achieve these features using a relatively small data footprint.

DISimS uses existing software modules that are re-engineered appropriately to achieve the design goals. The data storage and communication mechanisms ensure that there is no bottleneck due to large scale data movement. We also modified the simulation algorithms to incorporate new features, such as rollback of previously applied intervention strategies. This helps in achieving a high level of interactivity in real time along with the ability to get dynamic user input.

The software sub-systems that were part of the integration effort included Epifast [Bisset et al. 2009] – an HPC-based simulation engine, that simulates disease propagation process over a given region; ISIS – a web-based visual interface tool, that analyzes the role of different parameters in disease propagation; and a database repository, storing and operating on the demographic and geographic information about different regions and interactive data from users. We analyzed the feasibility of existing middleware platforms to support the integration and remodeled the middleware for supporting optimal data movement and interactivity.

DISimS is specifically designed to improve user experience. This includes user productivity and ease of use of the system. Care has been taken so that a user can access complicated mathematical models without having to become a computing expert. For instance, the environment provides automated services for experiment set-up and management, thus reducing the overall time of conducting end-to-end experimental studies. It also allows reuse of past epidemiological experiments and their results, which avoids duplication of efforts. We demonstrate the applicability of DISimS by describing two computational experiments. The experiments are motivated by real-world case policy studies and highlight the important features of DISimS.

2. BACKGROUND AND RELATED WORK

Computational models used in the study of epidemics comprise of two broad categories – aggregate based models and individual based networked models.

Aggregate differential equation based models partition individuals into separate classes such as Susceptible, Exposed, Infectious and Recovered (SEIR model), depending on the model of infection dynamics chosen. These models assume homogeneous mixing of populations and use differential equations to study the changes in epidemic states of populations over time. Some examples of aggregate differential equation based models used for studying disease propagation include [Rvachev and Longini 1985; Hufnagel et al. 2004]. An open source spatio-temporal epidemiological modeller system called STEM [Edlund and Kaufman 2012], developed by IBM in collaboration with Eclipse, Johns Hopkins University and others, also uses this approach.

Individual based networked models explicitly represent individuals within a population. The individuals interact via an interaction network. Recent research in this area by [Meyers 2007; Meyers and Dimitrov 2010; Pastor-Satorras and Vespignani 2002; Barrat et al. 2008; Newman et al. 2002] includes finding closed form analytical results on random graphs for finding epidemiological patterns of interest. Another type of individual based models use important statistics of a region such as density of individuals in a region using land scan data and basic census information
to get demographic distribution of individuals in a region, to model epidemic propagation. Research in this area includes work by [Germann et al. 2006; Ferguson et al. 2006; Ferguson et al. 2003]. Some researchers have also explored a hybrid approach where counties are represented as nodes and the movement of individuals represent the edges. Coupled rate simulations are used for propagation simulation between counties. Example of a high performance agent-based simulation based on this approach is the Global-Scale Agent Model (GSAM) [Parker and Epstein 2012]. It simulates propagation over billions of agents.

The last category of individual based models try to model individual level interaction between people based on their day to day activities and generate a graph of social contact network used for epidemic propagation. Research in this area includes work by [Keeling and Eames 2005; Meyers 2007; Barrett et al. 2008; Eubank 2002]. The back-end models used in the implementation of DISimS also broadly fit in this category.

The process of epidemic simulation modeling and implementation requires deep technical expertise for execution. Over the years, epidemiologists have worked in close collaboration with computer scientists to study the spatio-temporal diffusion of an epidemic. Traditionally, computational scientists received a set of epidemic scenarios to simulate from public health agencies such as DTRA (Defense Threat Reduction Agency) and CDC (Centers for Disease Control and Prevention). The scientists would model the scenarios and run simulations on high end computing resources such as clusters, clouds and grids. As new requirements came in, the simulation code was modified to accommodate new changes. When the experiment execution was completed, the experimental results were reported back to the health agencies for decision making in real world. This process could take weeks to months, depending on the complexity of the epidemic propagation and intervention scenarios. Moreover, there are infinite combinations of interventions based on type of policy decision and subpopulations to apply the interventions on. Hence, it was not only inefficient but also very difficult to implement all types of interventions embedded within the epidemic propagation process. This would also require constant changes to the high performance simulation code. To overcome this shortcoming, INDEMICS [Bisset et al. 2010] was developed as a data intensive, high performance modeling platform that uses database management systems for applying interventions externally to propagation engines like EpiFAST [Bisset et al. 2009]. INDEMICS uses a SQL based language called INDEMICS Query Language (IQL) to specify interventions through the INDEMICS client, which reduces the development effort significantly with minimum effect on efficiency. An example of an INDEMICS client script written in IQL can be found in the Appendix. INDEMICS could be directly accessed by public health decision makers by writing appropriate IQL scripts for applying intervention policies to the epidemic propagation process. However, INDEMICS still lacked ease of use required by public health decision makers to study epidemics in a productive way.

This paved the way for further research into ease of use and user productivity. The recent advances in computing technologies and ubiquitous computing have enabled access to remote computing resources easily. One such recent work also discusses providing access to high performance computing resources as a service[AbdelBaky et al. 2012]. There has been significant research in the area of accessing epidemic simulations through web based systems, such that simulation complexity is hidden from users. Research has also been conducted in the broader area of applying different visualization techniques for representing high end simulations on web based front-ends. Some examples of graph visualization techniques and tools developed include Gephi[Bastian et al. 2009] and Pajek[Batagelj and Mrvar 1998]. Easy accessibility through visualization is particularly important in public health epidemiology because domain specialists have limited technical expertise to execute and analyze complex simulations on high end computing platforms like clusters and grids.

Some examples of research conducted in developing web based systems and visualization platforms for epidemic simulations include Epinome[Livnat et al. 2012], Gryphon[Yu et al. 2010], FRED/Framework for Reconstructing Epidemiological Dynamics][Grefenstette et al. ], GLEaMviz[Broeck et al. 2011] and ISIS(Interface to Synthetic Information Systems). Epinome is a user centric system with visual analytics support for epidemiology that helps users in evaluation of strategies based on available information. Gryphon presents a modeling environment to represent geographic spread of the SARS outbreak, based on published data. FRED is an open source modeling system developed collaboratively by University of Pittsburgh and Carnegie Mellon University that captures interacting effects of migration strategies, behavioral changes of people and evolution of virus. GLEaMviz is a desktop based visualization and analytical software that simulates disease propagation based on integration of data at three levels - data on global population, data on population mobility and model of infection based on disease dynamics.

We developed ISIS based on extensive interactions with policy analysts over a 10 year period. Our early work in [Eubank et al. 2004] led to a realization that models can be made more useful only when they are easily accessible to end-users. ISIS allows users access to models such as EpiFAST [Bisset et al. 2009] and EPISIMDEMCIS [Barrett et al. 2008]. It allows selection and analysis based on a range of parameters that play an important role in epidemic propagation, such as disease models, efficacy of interventions, compliance rate and so on. ISIS also allows embedded management and storage of experiments that saves time to set up and manage epidemiological experiments.

ACM Transactions on Management Information Systems, Vol. V, No. N, Article A, Publication date: January YYYY.
Although useful, ISIS lacks features that facilitate interactive simulations. It also lacks a mechanism to incorporate real time data based on situation assessment for better evaluation and a provision to roll back previously applied interventions to evaluate current state of epidemic and apply new interventions. Addition of these features can lead to significant improvement in user-productivity and capability while executing epidemiological experiments. However, these features entail not just improvements to the web-based front-end systems, but also enhancements to the simulation models and platforms to support the level of communication and data movement involved. We present an architecture for developing complex interactive simulation environment for studying epidemics and their mitigation strategies, embedding all the necessary features for supporting improvements in user-productivity and capability.

Table I compares the features of existing epidemic systems with DISimS based on several design principles. Because of its unique architecture, DISimS is able to provide interactive computations and analytical visualizations along with simulation performance and efficiency.

3. ARCHITECTURE AND IMPLEMENTATION

In this section, we describe the architecture and implementation of DISimS. DISimS architecture can support user interactivity at periodic intervals such that users have the ability to pause the simulation at a specified time step and analyze the dynamics of the epidemic. The architecture also supports roll back of previously applied intervention strategies to contain the epidemic and allows users to apply new interventions based on evaluation of current epidemic state.

3.1. Architectural considerations

Currently, several well accepted technologies and paradigms are available for developing a software architecture of large-scale network-centric software systems. This includes service oriented architectures, distributed object architectures and resource oriented architectures. Many different technologies such as web services [Curbera et al. 2002], Jini [Waldo et al. 1999], REST [Fielding 2000] and so on provide specific instantiations of these architectures. However, application of a single architectural paradigm may not satisfy all the design requirements, namely usability, performance, interactivity and efficiency. This requires judiciously combining appropriate paradigms and technologies to produce an integrated modeling environment. Figure 1 shows the high level architecture of DISimS. In the remainder of the section we discuss various aspects of the architecture.

3.2. Component Systems of DISimS Architecture

At a minimum, an integrated epidemic modeling environment should consist of three broad components: (i) an epidemic propagation and intervention simulation engine, (ii) an easy to use user interface to interact with the backend, and (iii) a middleware to support communication and interactions between components. These components can be integrated in a variety of ways. For instance, a typical propagation simulation engine that simulates high resolution disease spread using individual based models supports relatively large scale data transfer rates to maintain performance and efficiency of execution. On the other hand, front end systems such as a web-based system, focus on a small amount of data taken as input from users to be passed to the backend. Depending on the level of interactivity with

<table>
<thead>
<tr>
<th>System Name</th>
<th>Web interface</th>
<th>User interactivity</th>
<th>Analytical Capability</th>
<th>Scalability</th>
</tr>
</thead>
<tbody>
<tr>
<td>FRED</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Large scale models</td>
</tr>
<tr>
<td>EPINOME</td>
<td>Yes</td>
<td>Partial</td>
<td>Yes</td>
<td>Pre-run simulated outbreak models</td>
</tr>
<tr>
<td>Gryphon</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Relatively smaller scale models</td>
</tr>
<tr>
<td>GSAM</td>
<td>No</td>
<td>No</td>
<td>Partial</td>
<td>Highly scalable with high resolution models</td>
</tr>
<tr>
<td>Flute</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Large scale individual-based models</td>
</tr>
<tr>
<td>GLEaMviz</td>
<td>Desktop based visualizations</td>
<td>Yes</td>
<td>Yes</td>
<td>Large scale hybrid models</td>
</tr>
<tr>
<td>DISimS</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Large scale high resolution models</td>
</tr>
</tbody>
</table>
users, the amount of data passed back and forth may differ. The architecture of the integrated modeling environment has to encompass these contrasting features and ensure optimal data movement and frequency.

For the purpose of developing an interactive simulation environment that could be productive for epidemiologists and public health decision makers in making policy decisions, we decided to reengineer some of our existing systems and integrate them to achieve our design goals. This resulted in substantial savings in terms of software development effort. The systems that have been integrated as part of DISimS architecture include

1. INDEMICS Epidemic Propagation Simulation Engine (IEPSE)
2. INDEMICS Intervention Simulation and Situation Assessment Engine (ISSAE)
3. Dynamic INDEMICS Client
4. ISIS Web Server and Web Client (IWEB)

As discussed in Section 2, INDEMICS moves the modeling of interventions external to the Epidemic Propagation Simulation Engine so that new interventions can be implemented without having to change the high performance simulation code of the propagation simulation engine. INDEMICS separates the task of intervention application in a separate module called the Intervention Simulation and Situation Assessment Engine (ISSAE).

In the DISimS architecture, IEPSE is implemented using the modified version of EpiFAST [Bisset et al. 2009], a simulation engine which can simulate disease spread in large-scale populations consisting of millions of individuals.

Table II. Components of DISimS, corresponding technologies and primary design concern

<table>
<thead>
<tr>
<th>Component system</th>
<th>Description</th>
<th>Implementation Technology</th>
<th>Design Concern</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISIS Web Client and Web Server</td>
<td>Web-based visual interface tool</td>
<td>Google Web Toolkit</td>
<td>Usability</td>
</tr>
<tr>
<td>INDEMICS Epidemic Propagation Simulation Engine</td>
<td>High performance simulation engine</td>
<td>C++/MPI</td>
<td>Performance and Efficiency</td>
</tr>
<tr>
<td>INDEMICS Intervention Simulation and Situation Assessment Engine</td>
<td>Intervention Selection and Situation Assessment engine</td>
<td>Oracle Relational DBMS</td>
<td>Flexibility</td>
</tr>
<tr>
<td>Dynamic INDEMICS Client</td>
<td>Intervention Subpopulation Selection Platform</td>
<td>Scripting language embedding SQL</td>
<td>Interactivity</td>
</tr>
</tbody>
</table>

ACM Transactions on Management Information Systems, Vol. V, No. N, Article A, Publication date: January YYYY.
EPIFAST was developed as an HPC based system using C++ and MPI and simulates the spread of epidemics through large populations, capturing the co-evolution of individual health, behavior, and disease transmission. It also has the capability to execute multiple replicates in order to capture the variability of the results due to the stochastic nature of the phenomena being modeled. The EPIFAST engine can execute the propagation process at a very rapid pace, simulating disease spread in a city like Miami, which has a population of about 2 million individuals, in less than 30 seconds, when executed on a 10 node cluster with 8 cores per node.

ISSAE is implemented using a relational database management system. INDEMICS provides a simple query language called the INDEMICS query language (IQL) as input to the ISSAE to specify interventions. The ISSAE stores demographic, social contact information and time varying infection data about individuals in the regions on which simulations are executed. Use of relational databases to represent ISSAE allows real-time information to be incorporated quickly into the system so that the users can perform situation assessment. The IEPSE component updates the database with information of infected individuals at each time step. This information can be retrieved using the IQL by running different queries to know the state of the system. Based on state assessment, a new subpopulation can be selected from the database for the next time step.

To provide web based access to the epidemic simulations represented above, we decided to extend ISIS Web Client and Web Server. ISIS was initially developed as a web-based front end for selecting initial parameters in the execution of simulation engines such as EPIFAST and EPISIMDEMICS. The parameters of execution that are taken as input from users include region of simulation, disease model, initial conditions, simulated days and intervention triggers and actions. Parameters related to intervention actions such as efficacy, compliance rate, availability and so on are also taken as input from users. See Figure 2 for a screen shot of the ISIS.

ISIS is developed using GWT (Google Web Toolkit), which provides a Java based platform for developing web systems. ISIS can be accessed by multiple users simultaneously from any machine with Internet connectivity using any standard web browser. It has a simple graphical user interface that makes it easy to use for public health decision makers and epidemiologists. Moreover, it can also generate different graphs based on results of the simulation run, which can be subsequently studied for analysis.

For supporting interactive features with users at specified time intervals, we extended the ISIS client so that users can execute a simulation for certain time interval and then pause it to analyze the effects of initial intervention strategies on disease propagation. This includes analysis with respect to different demographics and geo-spatial effects. The users have also been given the ability to roll back the effects of an intervention till a specified time period such that the system is restored to its previous state. The user can resume operations from that time step forward.

In the original INDEMICS architecture, since there was no web interface, the INDEMICS client scripts would be implemented by users using IQL and executed manually. In DISimS, since the details of execution are hidden from users behind the ISIS web interface, DISimS architecture allows creation of dynamic INDEMICS client scripts based on predefined templates and automates their execution on the compute resources. The creation of the scripts is handled in the middleware platform as described in the following section and is based on the parameters of execution selected by users from the ISIS web interface.

3.3. Middleware platforms to support data movement and interaction

The components of the DISimS architecture described in Section 3.2 operate at different levels of abstraction. The IWEB including the ISIS Web Client and ISIS Web Server interface represent data and operations at a higher level of abstraction for the convenience of end users. The ISIS Web Server, hosting ISIS web application and the HPC-based IEPSE simulation engine are typically not co-located on the same machine. Hence, the input parameters selected by
users through ISIS Web Client and passed to the ISIS Web server, need to be relayed to the IEPSE through some communication mechanism. The amount of data passed as parameters is usually small in scale. Once the propagation simulation starts at the IEPSE, it may run for a long time depending on the parameters selected. Hence the nature of communication between ISIS Web Server and IEPSE has to be asynchronous. Moreover, even with interactions, the IEPSE has to execute for some time before the user can analyze the effects of any intervention. Hence the frequency of communication between the ISIS Web Server and IEPSE is also not large. We evaluated the feasibility of existing middleware platforms for this parameter passing mechanism. One such middleware, “Simfrastructure” had been designed to support communication between ISIS and EPIFAST. In its older version, Simfrastructure included a mechanism to pass parameters for starting the epidemic propagation process in EPIFAST.

Simfrastructure was conceptualized as a high performance distributed system mechanism that could handle asynchronous communication between various service components distributed across different networks. Simfrastructure is based on tuple spaces based architecture with associative memory paradigm. It was not intended to support large scale data movement at regular intervals, but as a coordination framework to allow a consumer to request service fulfillment from a service provider. The main component of Simfrastructure that supports its service oriented model is the Blackboard, on which service requests are placed by the consumer. The other important component of Simfrastructure is called the Service Broker. Each Service Broker provides service fulfillment of a particular service that it is specialized to serve. The Service Broker continuously monitors the blackboard and when a service request placed by a consumer is found, it reads the request and fulfills it. The current version of Simfrastructure uses Jini, a service oriented architectural model of communication to implement the blackboard mechanism. Jini-based architecture [Waldo et al. 1999] provides a service oriented abstraction for registering and removing services easily on the fly. Each of the component systems of Jini can be considered as a composable service. Data can be written on to the Javaspace with a specific leasing time specified by the producer of data and any registered consumer service can access the data from the Javaspace. The producer and consumer systems need not be co-located on the same server and may be independent standalone systems or clusters.

We extended the Simfrastructure middleware platform (SMP), to support communication with INDEMICS system at the backend and for passing additional information to support interactivity and dynamic situation assessment on behalf of users. To enable communication with INDEMICS, we included a new service broker component called the INDEMICS broker inside Simfrastructure and added new client script templates within the database associated with the middleware. The INDEMICS broker acts as the main connecting component between the ISIS Web Server and INDEMICS. It not only allows to configure client scripts, but also maps the user parameters to their corresponding database equivalents. Based on the parameters selected by users using ISIS Web Client and passed using Simfrastructure middleware platform, the INDEMICS broker generates an INDEMICS client script dynamically, based on pre-defined script templates. This allows automation of script generation, such that the complexity of the IQL scripts is hidden behind the ISIS Web interface.

The other aspect of the architecture is to deal with large scale data communication between the IEPSE i.e. high performance simulation engine and the ISSAE, implemented using the relational database. The IEPSE executes the propagation simulation on high end computing resources such as clusters and grids and has high efficiency requirements. If ISSAE is directly connected to the IEPSE, then any change in the implementation of ISSAE would need changes to the high performance code of IEPSE. For making the system modular, flexible and adaptable, the IEPSE and ISSAE have to be connected through an optimized middleware platform. Since IEPSE executes a simulation over several time steps (typically represented as days), data has to be retrieved from the database and passed to IEPSE over many time steps. This data passed back and forth can be up to several megabytes. An orchestration mechanism of a service oriented architecture can slow the speeds of the IEPSE. Hence, a service oriented abstraction such as the one provided by Simfrastructure, where service requests are made asynchronously and with small data, cannot be used to

<table>
<thead>
<tr>
<th>Middleware Platform</th>
<th>Simfrastructure Middleware Platform</th>
<th>INDEMICS Middleware Platform</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Spaces-based middleware for high performance distributed systems between ISIS and HPC simulations</td>
<td>Java based middleware for data transfer between database system and high performance computing simulation system</td>
</tr>
<tr>
<td>Usage</td>
<td>Message-oriented data transfer</td>
<td>Large scale raw data transfer</td>
</tr>
<tr>
<td>Typical Comm. Data Size</td>
<td>140B per simulation</td>
<td>16KB per iteration of simulation</td>
</tr>
<tr>
<td>Design concern</td>
<td>Loose coupling and Modularity</td>
<td>Performance and efficiency</td>
</tr>
</tbody>
</table>

Table III. Middleware platforms and their speed comparison. Note: A single simulation experiment is carried out over multiple temporal iterations to study epidemic spread patterns over a period of time
As described in INDEMICS [Bisset et al. 2010], the INDEMICS middleware platform is composed of multiple modules including the Interpreter, Message deliverer and Session manager, which have been optimized for rapid data movements. For mapping to the database equivalent format, the Interpreter converts the database parameters into a suitable format and further hands it over to the message deliverer. The message deliverer then directly connects to the database and retrieves large data from the database. The Simulation session pool module maintains a queue of

connect the IEPSE and ISSAE. To serve our goals of efficiency and high data transfer rates, we decided to extend the INDEMICS Middleware Platform.

Fig. 3. Timing diagram showing the sequence of events that take place inside DISimS when a request for simulation execution is submitted at the beginning of the simulation or resumed from Paused state.
data retrieved from the database, to be passed to the IEPSE. The INDEMICS Middleware Platform (IMP) can support data transfer volumes of several megabytes over many continuous iterations. Instead of invoking static INDEMICS client scripts for interventions to IEPSE, the Interpreter module of INDEMICS Middleware Platform was modified to communicate with the dynamic client scripts generated by INDEMICS broker.

Using IMP as the additional middleware, large scale data volumes can be supported per time step across multiple simulation runs and thus the required performance of the DISimS simulation system is maintained. As can be seen in Figure 1, the presence of two middleware systems - SMP and IMP connected together in DISimS, instead of a single generic middleware, allows the simulation system to accomplish its usability goals along with performance.

Table III provides a comparison of the two middleware platforms - SMP and IMP, their main objectives, the data size that each of them support and the granularity of computation. As can be seen from the table, these two middleware have been specialized for different purposes and cannot be replaced by the other.

3.4. Data and context flow
In Sections 3.2 and 3.3, we described the architecture and implementation of DISimS through integration of different component systems and middleware platforms. In this section, we describe the data and context flow within DISimS as shown in Figure 4.

DISimS is accessible from any machine connected to the Internet through a web browser. Users, in particular epidemiologists and public policy decision makers, select different parameters of epidemic simulation execution to run an experiment through the ISIS Web Client. Parameters of execution include the region of study, number of days of simulation, replicates, disease model, initial conditions and so on. In addition, users can select intervention strategies to be applied to the propagation process. Interventions include vaccination, antiviral prophylaxis, antiviral treatment, social distancing and so on. Requests can also be submitted to perform some complex epidemiological experiments by applying interventions such as the Block-based intervention strategy. These interventions are specified as IQL scripts stored as templates. Refer Section 5 for definition of the Block-based intervention strategy and an example of an IQL script written for Block-based intervention. This information is passed as input to the interface broker of the Siminfrastructure middleware as a message. The interface broker interprets the data and submits a request for simulation to be executed on the blackboard.

ACM Transactions on Management Information Systems, Vol. V, No. N, Article A, Publication date: January YYYY.
The INDEMICS broker added as one of the sub-components of SMP, is one of the most important components of DISimS. The INDEMICS broker continually monitors the blackboard for new INDEMICS simulation requests. Once an INDEMICS service request embedded with the required parameters is found on the blackboard, it places a new execution request for starting the propagation simulation on IEPSE. The execution broker monitors execution requests on the blackboard and starts execution of propagation simulation on the available compute resource such as a cluster. The INDEMICS broker maintains several distinct client script templates for different intervention studies.

Based on the intervention parameters submitted on the blackboard through ISIS web client, the INDEMICS broker selects corresponding client script template and replaces the dummy parameters with the actual experiment and intervention parameters of execution. It also manipulates parameter such as the region name to a format which can be used by the execution broker for executing the simulation. The dynamically generated client script is in IQL that can be interpreted by the INDEMICS Middleware Platform. The IMP is configured as a background process that is always in a running state. Based on the invoked intervention (client) script, the IMP connects to appropriate database tables, retrieves intervened population data and passes it as an intervention to the IEPSE.

Similar to the execution broker that signals IEPSE to start execution, the Analysis broker starts the analysis script based on the analysis request made by a user. The request for analysis of a particular experiment is made to an Analysis server that runs the R statistical tool. The results of analysis are written on to the blackboard and consequently passed to the front end through the interface broker, where the corresponding graphs are plotted for the user. In Figure 4, the data and context flow of a simulation request is shown in steps 1 to 14, while that of an analysis request is shown with gray circles in steps 1 to 7.

The user may also decide to use DISimS interactively by pausing the simulation after a certain duration and running analysis on the simulation. Based on results of the analysis, the user may decide to apply a different intervention strategy for the rest of the simulation. In this case, the first client script gets executed as described above. When the simulation stops, and user can choose to run analysis, by submitting an analysis request on the blackboard through the ISIS Web Client, and may view the results as a plot from within ISIS. When the second intervention strategy is selected for the rest of the simulation, a new request is generated and new parameters are passed back to the backend through the blackboard of SMP. The INDEMICS broker invokes a new client script starting from the day the simulation was paused. This new client script connects to the same session with the INDEMICS Middleware Platform. The IMP also holds its connection with the IEPSE and sends new interventions based on externally selected subpopulation data from ISSAE. The timing diagram shown in Figure 3 displays the exact sequence of events that take place inside DISimS when a user submits a new simulation request or resumes a previously paused simulation experiment.

3.5. Data Abstractions and Specification

In this section, we describe the internal data specification and formats for data interchange used in DISimS in a formal way. The data representation and abstractions have been designed to ensure minimum amount of data transfer and optimal frequency of data exchange between the modules, in order to satisfy requirements of efficiency and performance.

3.5.1. ISIS Web Client and Server. As described before, users select parameters of simulation through the ISIS Web based interface. Users also select the type of intervention strategies to reduce the effects or intensity of epidemic propagation. The data selected through ISIS Web Client that is passed back to INDEMICS at the backend through ISIS Web Server can be represented as a vector \( \text{simtype}, \text{param}, \text{intv}^* \) where

- \( \text{simtype} \) is the type of propagation simulation that is to be executed at the back end such as INDEMICS, EpiFAST and EpiSimDEMCIS.
- \( \text{param} \) is the set of parameters that represent the epidemic simulation including region, disease model, initial conditions and so on.
- \( \text{intv} \) is a set of one or more intervention strategies that can be applied.

\( \text{param} \) is a vector given by \( \text{region}, \text{dismodel}, \text{initial}, \text{simdays} \), where

- \( \text{region} \) is the region on which the epidemic simulation has to be executed.
- \( \text{dismodel} \) represents the disease model such as catastrophic flu, mild flu, H1N1, H1N2 and so on.
- \( \text{initial} \) represents the initial conditions of the epidemic
- \( \text{simdays} \) represents the total number of simulated days

\( \text{intv} \) is a vector given by \( \text{subpop}, \text{intvAction} \), where \( \text{subpop} \) represents the subpopulation on which the interventions have to be applied and \( \text{intvAction} \) is a vector given by \( \text{type}, \text{del}, \text{eff}, \text{dur}, \text{compl} \) where

- \( \text{type} \) is the type of intervention to be applied such as vaccination, social distancing, anti-viral.
— \(del\) represents the delay in implementing the intervention action in real world.
— \(dur, eff\) and \(compl\) represent the duration, efficacy and compliance rate respectively of the intervention action applied on the targeted population.

The data represented by the vector \((\text{simtype}, \text{param}, \text{intv})\) is passed from the ISIS web server to the Interface broker of SMP. Based on the selections made by the user, a request is submitted on the blackboard to start a propagation simulation of type \(\text{simtype}\) at the back end. The INDEMICs broker reads the request and the parameters along with the intervention strategy. It places a request to start the propagation simulation of type \(\text{simtype}\) and the execution broker starts the simulation.

Intervention strategy - \(\text{intv}\) is used for creating the client script from the appropriate INDEMICs client template. The template parameters are replaced by actual values represented by objects of vector \(\text{intv}\). Objects of \(\text{param}\) are also used to replace template parameters as values in the dynamic INDEMICs client script.

Whenever an interactive simulation is paused by the user after a fixed duration, the user can request an analysis to be run on the intermediate results. The output from the analysis is displayed to the user in the form of graphs. Based on the outcome of the analysis, the user might decide to submit new intervention strategies for the next duration. This new data is also represented similar to the original vector \((\text{simtype}, \text{param}, \text{intv})\) and is passed to the Interface broker of SMP.

3.5.2. Propagation simulation. The propagation simulation, represented by the IEPSE component of INDEMICs is a concrete implementation of the extended CGDDS framework described in the paper [Ma et al. 2011]. The primary input is a social contact network, that represents proximity relationships between individuals of the population. It is represented by the Graph \(G(V, E)\), where \(V\) is set of vertices representing the individuals of the population and \(E\) represents contact between them. Each vertex \(v \in V\) has an associated vector \(V = (\text{pid}, h, t_1, t_2, l_1)\) where

— \(\text{pid}\) is the person identifier for the given vertex,
— \(h\) is the health state based on SEIR model,
— \(t_1\) is the time at which the vertex is infected,
— \(t_2\) is the time of recovery and
— \(l_1\) is the list of interventions applied on the vertex.

Each edge \(E\) is represented as a vector \(E = (V_1, V_2, p)\) where

— \(V_1\) and \(V_2\) are the vertices on which the edge is incident and
— \(p\) is the probability of transmission between the vertices as defined in the propagation algorithm.

IEPSE reads the entire graph \(G(V, E)\) into the main memory from a flat file at the beginning of the simulation, based on the value of the parameter \(\text{region}\), selected by the user through ISIS. \(G(V, E)\) remains unchanged throughout the simulation. Interventions change the edge attributes and can simulate edge deletion by using appropriate edge label.

The other input to IEPSE is the list of interventions selected by users through the ISIS interface and retrieved from the RDBMS based Situation Assessment Engine. This is given by the vector \(\text{intv}\) - \((\text{subpop}, \text{intvAction})\). For implementation inside IEPSE, interventions are represented as \(I = (\text{pid}, \text{A})\), where

— \(\text{pid}\) represents the identifier of the person to be intervened and
— \(\text{A}\) represents the intervention action to be implemented. \(\text{A}\) is a vector given by \((\text{type}, \text{del}, \text{eff}, \text{dur}, \text{compl})\), where
   — \(\text{type}\) is the type of intervention to be applied such as vaccination, social distancing, anti-viral,
   — \(\text{del}\) represents the delay in implementing the intervention action in real world, and
   — \(\text{dur}, \text{eff}\) and \(\text{compl}\) represent the duration, efficacy and compliance rate respectively of the intervention action applied on the targeted population.

After the intervention \(I\) is obtained from ISSAE through INDEMICs middleware as input at every time step, IEPSE runs propagation simulation and computes disease propagation in the form of a list of individuals infected in the next time step.

The output from IEPSE is a vector \(O\) of the form \((\text{infected}, \text{infector}, \text{infDur}, \text{diagnosed}, \text{incDur})\) where

— \(\text{infected}\) represents the set of newly infected vertices in the current time step,
— \(\text{infector}\) represents the corresponding vertex identifiers that infected them,
— \(\text{infDur}\) is the duration for which the vertex would remain in the Infectious disease state,
— \(\text{diagnosed}\) are the vertex identifiers that are not yet symptomatic and
— \(\text{incDur}\) is the period for which the diagnosed vertices remain asymptomatic.
The output from IEPSE is passed to ISSAE through the INDEMICS middleware at every time step. Since the data is passed back every time step and also the output contains a set of PIDs as given above, the scale and frequency of data transfer is large, which requires a specialized middleware like the IMP.

Whenever a user requests a simulation to be paused after a certain duration to be able to analyze the results of the intervention strategy applied till that point, then IEPSE holds the connection with IMP just like it is expecting data for the next iteration. When a new INDEMICS script is generated through the SMP it is passed to IMP and IMP makes the necessary data conversions to pass data to IEPSE without having to make any other changes.

3.5.3. INDEMICS Intervention Simulation and Situation Assessment Engine (ISSAE). The ISSAE component of DISimS is based on relational database management systems. The IMP converts the dynamic client script received from the INDEMICS broker component of SMP and transforms the IQL statements from the code in pure relational queries that can be interpreted by the relational database representation of ISSAE. ISSAE stores and processes four kinds of datasets: the social contact network data $N$, demographic data $R$, infection dendogram data $D$ and intervention data $I_n$.

Demographic data for each region is stored in a simple relational format in a table given by the tuple $R = (pid, age, gender, income)$. $R$ is static and remains unchanged for the duration of a simulation. New demographic value sets can be added to the tuple based on availability of information for the population.

The social contact network data $N$ is stored as a tuple $N = (pid_1, pid_2)$, where $pid_1$ and $pid_2$ represent the end points of an edge in the social contact network. This is a copy of the data used by IEPSE to simulate epidemic propagation. It is stored in the RDBMS so that interventions based on social contact network structure can be formulated.

The temporal data related to infections is stored in a separate table which can be directly updated based on the output received from IEPSE. This data can be represented by the tuple $T = (\text{infected, infector, infDur})$ where

- $\text{infected}$ represents the set of newly infected vertices in the current time step,
- $\text{infector}$ represents the corresponding set of vertices that infected them and
- $\text{infDur}$ is the duration for which the vertex would remain in the infectious disease state.

ISSAE is used to support situation assessment and intervention simulation. The output obtained from ISSAE is given by $I(S', A)$ where the set $S'$ contains person identifiers on whom to apply interventions and $A$ represents the intervention actions. IMP sends this to IESPE. When IEPSE resumes computation, it uses the new vertex and edge labels to evaluate the epidemic propagation for the next time step.

In case of interactive computations, the IMP receives new INDEMICS intervention script and converts the IQL to relational format same as before and gets new data from ISSAE and signals IEPSE to resume operation for next time step with new data.

4. COMPUTATIONAL EXPERIMENTS

In this section we illustrate the capability of DISimS via two computational experiments. The first one is a real world case study to evaluate the effectiveness of school closure intervention in containing an ongoing epidemic with DISimS. The second one is to illustrate user online optimization of intervention strategies along a decision tree via interactions with a DISimS-run simulation.

4.1. Computational Experiment 1

We present a computational experiment based on a realistic use case scenario. In this study, we evaluated the school closure intervention in a catastrophic flu epidemic. School closure has been deemed as an effective measure to contain a flu pandemic. For example, during the 2009 H1N1 flu outbreak, New York city officials ordered the closure of 30 schools “after an increase of reports of students with flu-like symptoms” [CNN 2009]. School closure reduces the overall transmission within a school and is a well known and effective non-pharmaceutical intervention [Cauchemez et al. 2009; Wu et al. 2010].

<table>
<thead>
<tr>
<th>Case study</th>
<th>System Name</th>
<th>Development time</th>
<th>Analysis set-up time</th>
<th>Expt. set-up+analysis time</th>
<th>Expt. execution time</th>
</tr>
</thead>
<tbody>
<tr>
<td>School</td>
<td>DISimS</td>
<td>5 minutes</td>
<td>20-25 mins</td>
<td>30 min</td>
<td>3-4 min/iteration</td>
</tr>
<tr>
<td>Closure</td>
<td>INDEMICS</td>
<td>1 day</td>
<td>0.5 day</td>
<td>1.5 days</td>
<td>3-4 min/iteration</td>
</tr>
<tr>
<td>intervention</td>
<td>EPIFAST</td>
<td>hard to implement</td>
<td>unknown</td>
<td>unknown</td>
<td>unknown</td>
</tr>
</tbody>
</table>

Table IV. Comparison of efforts for school closure intervention case study on Miami city. Experiment execution is carried out over multiple iteration days. *For DISimS, the expt. execution time excludes communication time between the web-server and INDEMICS Middleware Platform, which is negligible compared to actual expt. execution.

ACM Transactions on Management Information Systems, Vol. V, No. N, Article A, Publication date: January YYYY.
Against the potential risk of within school flu outbreak if schools remain open, are the large social costs associated with school closures. Later in the 2009 H1N1 pandemic, the Centers for Disease Control and Prevention (CDC) revised its recommendation from shutting down schools immediately when students became ill to keeping schools open even with flu outbreaks [Time 2009].

The school closure policy works as follows: if in a school, the fraction of students diagnosed with flu exceeds a certain threshold then the school is closed for a certain number of days, and for each diagnosed student below a certain age, one parent or care giver must stay at home. School closure is not a binary decision, but based on a number of parameters. We analyzed an event-triggered school closure policy for an experimental study on a flu outbreak in Miami, to assist analysts in their decision making in the real world. We considered two parameters in this measure: threshold and duration. The former determines the severity of the epidemic to make it necessary to close schools; the latter determines how long schools need to be closed. We chose two values for each of the two parameters to form four configurations of the school closure intervention. In contrast to previous studies where we examined the course of epidemic dynamics, the main objective of this study was to study and provide an effective comparison between a variety settings of an intervention policy. We point out that the study provides an illustration of possible interventions that can be easily simulated by DISimS but difficult for either EPiFAST or INDEMICS.

The complexity of the school closure intervention comes from two aspects: each school is determined to be closed individually instead of universal closures; for the affected subpopulation (students of the closed school) we need to identify another subpopulation consisting of people of appropriate demographic properties (age and household), who represent the care givers of the children.

When we were requested to perform the school closure study, our simulation engines such as EPiSIMDEMICS and EPiFAST did not have features to integrate supplemental information at that time and hence could only simulate simplified but less realistic versions of the school closure intervention. The code of these high performance simulation engines would have had to be modified to support such interventions. Our simulation engine developers and the intervention experiment strategy designers would have had to work together to precisely interpret the strategies and code them into the high performance engines. The estimated development time including requirements gathering, implementation, and testing would be several weeks, in contrast to the estimated experiment execution time of only one week. This approach would have been time-consuming and it would have been difficult to report simulation results and make policy recommendations in time.

To overcome this problem, we used INDEMICS, a database supported epidemic simulation framework, to run the study. In contrast to the epidemic simulation engine like EPiFAST, the implementation of interventions in INDEMICS is modeled by data query algebra, and the interventions are completely computed using query language of database management systems. Experiment strategy designers only need to describe their scenarios in IQL and submit the simulation jobs to INDEMICS for execution. The experiment development process of INDEMICS takes a few days to map the interventions into IQL. INDEMICS incurs marginal execution time overhead, but it needs no significant code development or testing. We adopted this solution to run the study and it greatly reduced the study period and saved a significant human effort.

Although the development time for implementing the intervention was shortened remarkably by INDEMICS, INDEMICS did not have a module to automatically set up experiments, monitor the state of an experiment and manage experimental inputs and results. There was no provision for reusability and sharing by checking if an appropriate INDEMICS intervention script was previously written by some other user. Also, when the intervention had to be simulated with different parameter settings, using INDEMICS became cumbersome. For example, a script to run a factorial experiment by changing multiple parameter values had to be prepared manually, which was error-prone. The simulation inputs and outputs had to be well organized to avoid overwriting or misreading. The simulation jobs also had to be monitored by the experiment executors. Such tasks needed considerable manual effort. Reading and understanding raw simulation results was difficult since INDEMICS did not have statistical analysis or data visualization modules. We realized that when the user has minimum knowledge and experience in preparing INDEMICS scripts (in IQL) and running simulations in a high-performance computing environment, which is often true for public health domain experts, executing complex experimental studies such as the one to implement school closure intervention, is difficult even with INDEMICS.

From the experience of implementing the School closure intervention study, we realized that the usability of the simulation system had to be leveraged further. Hence we developed DISimS, with features like user interactivity, simple interface, experiment data management, job monitoring and analysis in addition to the attributes that were already provided by INDEMICS and EPiFAST. Employing DISimS for the school closure intervention study could have reduced the overall experiment set-up and management time and enhanced the human productivity considerably. Using DISimS, the users now only need to select the intervention scripts and parameters of execution using an intuitive web-based graphical interface. The data files for the factorial experiments are well-organized and well-archived.
and the simulation jobs are automatically monitored and scheduled. **DISimS** introduces a marginal overhead of execution as compared to **INDEMICS**, which is equivalent to the communication time between the web based front end to the **INDEMICS** server middleware. Table IV shows the comparison of the efforts for the school closure intervention experiments using **EpiFAST**, **INDEMICS** and **DISimS** on the city of Miami. As can be seen in the table, the total human effort for experiment design and analysis is reduced significantly by **DISimS** compared to previous systems, and the total increase in the experiment execution time is negligible. This table shows the value of **DISimS** for improving the productivity of epidemiologists and public policy decision makers. They can now set up, manage, and execute complex intervention case studies without much help from the computational scientists.

Figure 5 shows epicurves in different intervention settings. It is an example of visualization that user can obtain directly from the **DISimS** system. In this plot, we can see that applying school closure too early (with 1% threshold)
may suppress the disease outbreak temporarily but the epidemic takes off soon. But it indeed postpones the epidemic peak; and the gained time may be useful for taking other measures.

4.2. Computational Experiment 2

The ability to interact with DISimS can be useful for training public health decision makers in evaluating the effectiveness of various intervention strategies. DISimS provides a specific type of interactivity while executing realistic large scale simulations. This interactivity allows users to pause a simulation after a specific duration, analyze the evolution of the epidemic in the current scenario and then come up with a new strategy that can work best for containing the epidemic in the given scenario.

At every point where the simulation is paused, the analyst has to make a decision about which intervention strategy to apply for the next duration ranging across several time steps, so that the epidemic can be contained effectively. This decision is based on situation assessment of the epidemic dynamics at the current time step, which is made possible because of the ISSAE component of DISimS. The various choices available to the analyst for decision making and the actions taken, which could affect availability of future choices, can be represented in the form of a decision tree. This decision tree is based on the experimental protocol of information available to users and the range of intervention actions that can be executed. The experimental protocol of an adaptive epidemiological experiment is shown in Figure 7.

**Experiment design.** In this computational experiment, we study a strong flu epidemic in the Montgomery County of Virginia. The county has a population of about 75,000 people. The public health decision maker has antivirals to distribute but the supply is limited. Starting from the beginning of the epidemic, every 25 days, 1,000 more units of antivirals become available. This is to address the limited pharmaceutical manufacturing capacities and latency of massive distribution of drugs. The antivirals are effective for 14 days once applied and they reduce probability of getting infected (for healthy people) or probability of infecting others (for ill people) by 80%.

A group of experts are chosen to participate in the experiment. Each can ask DISimS questions about the current epidemic dynamics and can decide how to distribute the current antiviral supply based on the answers to the questions. For illustration, we pause the simulation every 25 time steps (25 simulation days) to allow the experts to query and intervene based on age groups and current health state of each person in the Montgomery County. We point out,
however, that the simulation can be paused at any time step and the user interaction can be based on any data available from the simulation or in the database.

A public health analyst, at each decision point, can decide to do nothing, or apply $K$ units of antivirals, where $K$ is bounded by the current antiviral supply, to people randomly chosen in certain age groups. The possible decisions and the random epidemic trajectory formulates a stochastic decision tree, where at each decision point an analyst may decide to take different branches depending on the epidemic dynamics.

We ran this scenario with a public health analyst. Figure 8 shows the representation of the decision tree available to the analyst at the various decision points that he paused the simulation. In this case, the analyst paused the simulation on day 25 and day 50 respectively. The figure also shows the corresponding choices available for intervention application. For the sake of simplicity of the experiment, we made only limited choices available to the analyst for situation assessment and intervention application based on age groups. However, in reality the analyst can choose from a large range of information for situation assessment and intervention application, based on the Experiment protocol described in Figure 7.

Figure 6 shows epidemic curves generated from the DISimS system for the experiment. In the non-Adaptive case, antivirals are evenly distributed among each age group - School age, Adults and Senior Citizens on day 25 and day 50. In the adaptive case, antivirals are distributed based on situation assessment on day 25 and day 50. Since it was observed by the analyst that higher percentage of school children were infected in the first few days compared to other age groups, greater percentage of AVs were distributed to school children. As can be seen in the Figure 6, the peak time of the epidemic shifted to the right in the non-adaptive and adaptive cases and the peak infection count also went down. There is not much difference observed in the epi curves plotted for non adaptive vs. adaptive cases, since there is only a small difference in the percentage distribution of antivirals in both cases. Hence, this may not cause a significant difference to the epidemic dynamics. However, if the adaptive technique is used for training the analysts such that they can infer more about the epidemic dynamics and apply intervention actions at the right decision points, then there is a possibility of seeing change in propagation dynamics of the epidemic.

This experiment illustrates the following capabilities of the DISimS system: (i) A user can pause and resume a simulation at any time step. (ii) A user can interact with the simulation online, query the system and make decisions accordingly. DISimS enables adaptive interventions that address both uncertainty of epidemics and that of the human decision process. (iii) DISimS supports realistic scenarios of public health level decision making. (iv) DISimS supports simulations of realistic, implementable but complex intervention strategies, based on dynamic epidemic data, as well as demographic and other data.

5. CONCLUSION

In this paper, we have presented the architecture and implementation details of DISimS - an integrated, high performance computing oriented epidemic modeling environment. DISimS focuses on three important user level goals: (i) improving user productivity and ease of use, (ii) supporting interactive modeling and (iii) pervasive web-based access to the models. We described two computational experiments that illustrate the applicability of DISimS for addressing practical public health policy questions. We hope to deploy a prototype of DISimS for use by a broader scientific community in the near future. DISimS can be extended in several different directions. We mention three important ones here: (i) extending DISimS to further support optimal dynamic interventions, (ii) an evolved scripting language that allows a user to specify broad classes of situation assessment queries, and (iii) allowing multiple stake holders to interact with a single instance of the system to support distributed decision making.

More generally, modeling environments to study other socio-technical and biological systems, including urban transport systems, telecommunication systems and immune modeling systems, often exhibit similar modeling requirements as discussed here for epidemiological modeling environments. We believe that DISimS and its extension can thus applicability beyond the specific application domain described in this paper.

We discuss a specific extension that we are currently investigating – development of an educational gaming technology based on the epidemic modeling environment. Gaming systems not only need fast computations at run-time along with changing parameters, but also very detailed graphical interfaces for the convenience of users. Performance is a key requirement for such gaming systems along with a high resolution user-friendly graphical interface to keep the attention of the users.

REFERENCES


INDEMICS Intervention Simulation and Situation Assessment Engine (ISSAE).

We also briefly describe the INDEMICS Intervention Language (IQL) used to specify the intervention strategies in

APPENDIX

In this section, we describe some of the terms used extensively in the paper that need more detailed explanation. We also briefly describe the INDEMICS Intervention Language (IQL) used to specify the intervention strategies in INDEMICS Intervention Simulation and Situation Assessment Engine (ISSAE).
A.1. Block-based intervention strategy

Block-based intervention: This mitigation strategy specifies that if a fraction of people diagnosed with a disease in a census block exceeds a certain threshold, then quarantine or give medical treatment to individuals in the entire block. This strategy is often used by epidemiologists and public health decision makers when several cases of an outbreak are reported in a certain census block.

Algorithm 1: INDEMICS Client Script template of Block-based intervention using IQL

```plaintext
connect session: session_name = block-based-intervention
day from 0 to last_simulation_day do
    copy data: to set ISISTMP, from select b.block, count(b.pid) as total_diagnosed, day from BLOCK b, DIAGNOSED_PERSON d where diagnosed_time =day and b.pid = d.pid group by b.block
    define set: infected_block_day as select block from ISISTMP where day = day and total_diagnosed/total_persons >0.02
    update set: update INTERVENED_BLOCK set intervened = day where block in infected_block_day
    define set: intervened_population_day as select pid from PERSON_HOME_LOCATION where home_location in (select block from INTERVENED_BLOCK where intervened = day)
    set interventions: action= StayHome, compliance=1.0, duration=7 days, delay=0 days on intervened_population_day
stop session:
stop client:
```

A.2. INDEMICS Query Language (IQL)

The INDEMICS Query Language is used to fetch data related to interventions from ISSAE and use it to apply interventions to propagation simulation in IEPSE. The commands available in INDEMICS Query Language can be classified into two types: Simulation Setting/Cleaning Commands (SSCs) and Simulation Interaction Commands (SICs). As shown in the given example, connection session, stop session and stop client are SSCs; define set, update set and set interventions are SICs.

Both SSC and SIC are composed of two parts: the command operator and its parameter. For the SSCs, the command parameter is very simple. For example, the parameter of "connection session" command is the session name, which is the name of the session that the client is supposed to interact with. (Since INDEMICS client and IEPSE are distributed modules, they need to be connected first before further interaction). The situation assessment and intervention strategies are specified by set operators in INDEMICS Query Language. The command parameters of SIC are the data sets to apply set operators on. If relational database is used to implement ISSAE in INDEMICS, the data sets in the SIC command parameters represent tables in the database. Client may define a set, copy data to the set and update an existing data set. What operation to perform on the data sets is specified in the command parameters. In the current version of INDEMICS Query Language (IQL), we borrow much of the syntax from the structured query language (SQL) for data set operations. For example, in the given client script, data is copied from existing data sets to a target data set by applying filters similar to that in SQL.

An example client template script for a block-based intervention written in INDEMICS Query Language is given here. The SMP replaces the dummy variables with actual parameters creating a dynamic INDEMICS client script at run-time.