Middleware Solutions for Integrated Simulation Environments

Leila Jalali
Department of Computer Science
University of California, Irvine
CA 92697 USA
jalalil@uci.edu

Sharad Mehrotra
Department of Computer Science
University of California, Irvine
CA 92697 USA
sharad@ics.uci.edu

Nalini Venkatasubramanian
Department of Computer Science
University of California, Irvine
CA 92697 USA
nalini@ics.uci.edu

ABSTRACT

This paper outlines the expected research contribution of my PhD work. RAISE is a research project aimed at building a framework and platform that supports the integration of multiple existing models, simulations, and data. One of the limitations of the simulators is that they are developed by domain experts who have an in-depth understanding of the phenomena being modelled and typically designed to be executed and evaluated independently. Therefore the grand challenge is to facilitate the process of pulling all of these independently created models together into an interoperating systems of systems model where decision makers can explore different alternatives and conduct low cost experiments in an interactive environment. The key research question is whether such integration of independently created, deep domain models can be made feasible, practical, flexible, cost-effective, attractive, and usable.

Categories and Subject Descriptors

D.2.11 [**Software Engineering**]: Software Architectures Patterns (Reflection)

General Terms

Management, Design, Experimentation, Performance.

Keywords

Reflective Middleware, Simulation Integration, Structural Reflection, Metamodel.

1. MOTIVATION

Modeling and simulation is an important methodology for addressing a variety of real-world problems; it offers numerous advantages instead of experimenting with the real system itself. Simulation is cheaper, quicker, and enables what-if analyses for better system design [1, 8]. This paper outlines the expected research contribution of my PhD work. RAISE is a research project aimed at building a framework and platform that supports the integration of multiple existing models, simulations, and data. Building complex simulations to understand the joint effect of multiple phenomena (spread of hazardous material as a result of an earthquake, impact of obesity on health issues in a society) is very useful. For example in domains such as emergency response where response plans and methods are validated by simulating disasters and their impact on people and infrastructure. A variety of simulators have been developed in the domain of emergency response, e.g., loss estimation tools (HAZUS [13], INLET [14], CAPARS [21], CFD [22]), fire simulators (CFAST [10], CCFM [19]), evacuation simulators (Drillsim [9], SDSS [25]), transportation simulators (VISISM [18], PARAMICS [20]) etc. Since these simulators are typically developed by domain experts who have an in-depth understanding of the phenomena being modelled and their evolution (e.g. hazardous materials spread via plumes), they are designed to be executed and evaluated independently. Consider a fire simulator, CFAST, that simulates the impact of fire and smoke in a specific region and calculates the evolving distribution of smoke. Since fire and smoke can affect health conditions of individuals in the region of fire, one may wish to study its impact on the evacuation process as captured within an evacuation simulator, e.g. Drillsim. Similarly, the progress of fire (captured by CFAST) may create infeasible paths/exits for evacuation (as captured by Drillsim). An integrated and concurrent execution of the two simulators is essential to understand the adverse impacts caused such as increase in evacuation times or increased exposure to undesired particulates. Such what-if analyses can enable intelligent decision making to improve the outcome of the response.

Another example is health care systems in which understanding complex health issues requires combining multiple simulation models (including food systems such as agriculture, processing, transportation, distribution, climate, and social, economic, and physical systems in which people and food operate) to project the effects of policy choices into the future [27, 28]. Considering each constituent service system separately is a losing proposition. The same argument will also apply to urban planning systems and management of urban lifelines and infrastructure. The grand challenge in our research is to facilitate the process of pulling all of these independently created models together.

In our approach, RAISE, we explore reflective middleware solutions to address challenges of integrated simulation environments in which interoperability of different simulators can be ultimately achieved in a flexible and efficient manner [17]. In this framework, we extract relevant information from individual simulators using metalevel models, i.e. shared metadata. Meta models allow us to capture the properties of the different simulators, specify semantic constraints on how information should be shared across simulators and consequently reason about the interactions among them.

This paper proceeds as follows. In Section 2, we discuss some background efforts in the area of simulation integration, our reflective architecture, RAISE, and the main challenges in our research. In Section 3 we discuss system prototype implementation and evaluation. We use a detailed case study that integrates multiple real world simulators to evaluate our proposed approach and techniques. Finally we draw conclusions.

2. USING REFLECTIVE MIDDLEWARE FOR SIMULATION INTEGRATION

To best of our knowledge, simulation integration has been studied in two domains – (a) military command-and-control and (b) games and virtual environments. The U.S. Department of Defense (DoD) has promoted the development of distributed simulation standards to provide a common framework in which simulators can be integrated. These include standards such as SIMulator NETworking (SIMNET) [2], Distributed Interactive Simulation (DIS) [11], Aggregate Level Simulation Protocol (ALSP) [4], High Level Architecture (HLA) [26]. These standards provide specific services for interoperability in niche applications, for example DIS for human-in-the-loop simulators or ALSP for war games. The recent HLA effort has become the defacto standard technical architecture for military simulations in the United States – it aims to promote interoperability, scalability, and reusability between simulators. One of the central components of HLA is the Runtime Infrastructure (RTI); ORBs and CORBA services are candidate tools for implementing HLA RTIs. While HLA led to some new insights in simulation integration, its broader applicability for general simulation integration is questionable. It is a complex standard designed primarily for the military domain and is not transparent enough – too much low level knowledge is needed from the practitioner. Additionally it requires the participants to agree on a common interpretation of data that is produced and exchanged between them.

Recently simulation integration methods have been also used in the game community [5, 6, 7], primarily to support interoperability. As in the case with the HLA architecture, solutions here are prescriptive - they force the developers to provide a particular functionality or to conform to specific standards to participate in the integration process and have different assumptions/limitations on how participants interact. Such methods are unsuited to the integration of pre-existing (third-party) simulators since they often require modification of the original source code.

2.1 Reflective Middleware Approach

We propose a reflective middleware architecture, RAISE, (figure 1) to support the development of integrated simulation platforms. Our initial efforts [17] focused on using structural reflection [12, 3] to reify (abstract out) the structure of objects and components of the underlying simulators. The base-level consists of the various (pre-existing) simulators that must be integrated. In the proposed architecture, integration of different simulators can be ultimately achieved by using the meta-level for specifying/modeling the properties of the different simulators and reasoning about the interactions among the different simulators – i.e. what we intend to design and develop is a meta-simulator. The meta-level is built on base-level simulators; reification of base-level entities yield data structures at the meta-level, modified features of these structures that implement the integration are then reflected to the base-level. A closer look at the base-level simulators themselves reveals that the structural aspects of the simulation application are not merely in the simulator code, backend databases and models stored in domain-specific formats contain aspects of the simulators that may need to be explored as well, in general, there can be many kinds of meta-level entities to cover various integration aspects. By using the metamodeling capability the model elements that need to be integrated can be extracted. In other words in our approach, we formulate the metamodel that captures concepts of interest using a publish-subscribe mechanism for data exchange – here, subscribers (the simulation integration tasks) express interest in aspects that they want to observe (implemented by base-level simulators) – when changes in these monitored aspects occur at the base simulators, the meta-level entities receive information or updates of interest via publishers. A pre-existing set of ontology models assist in the matching process for the pub-sub implementation of the simulation integration task; these include domain ontologies that are representations of knowledge in a well-circumscribed domain. Interoperability of different simulators can be achieved by sharing and understanding the metamodels. Implementing the semantic constraints for simulation integration is a human in the loop process which results in the annotations that are invisible to base-level computation and are provided to the meta-level.

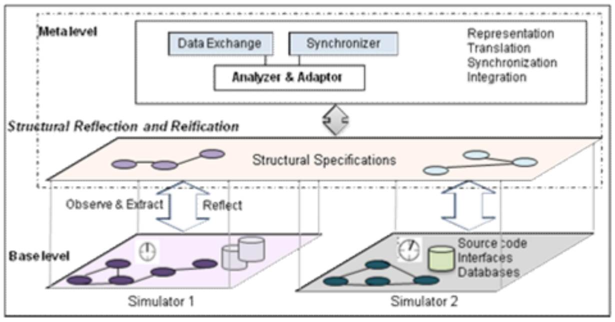


Figure 1. Reflective Middleware Architecture for Integrated Simulation Environments

In contrast to prior work on simulation integration, in RAISE we do not need to integrate simulations tightly into a common framework, but we make it feasible to semi-automatically compose simulation models via a looser coupling approach that avoids the need to adhere to a rigid common interface, which can hinder leveraging prior work. We explore a reflective architecture to address challenges of integrated simulation environments in which interoperability of different simulators can be ultimately achieved in a flexible and efficient manner while preserving the autonomy of the individual simulators.

2.2 Challenges

There are several challenges in our research project. The first challenge arises from modeling complex scenarios using multiple simulation models and the analysis of cause-effect relationships between those models. Given the potential black-box nature of simulators developed by experts in diverse domains we believe that achieving a completely automated plug-and-play integration of simulators is a very difficult, if not infeasible challenge. Our goals are more modest – we intend to develop enabling tools that will simplify the task of simulation integration with a wide range of simulators that vary in the degree to which they expose their interfaces and implementations. We do not want to require simulator developers to adhere to a strict programming interface or conform to particular design styles - the ability to flexibly interoperate with multiple simulators is our goal. Another challenge arises from the fact that each simulator uses its own models and entities; these must now be integrated in the context of a single simulation. The simulators need to exchange the data and have a correct interpretation of the data they send and receive. Time synchronization is yet another challenge. When integrating simulators, there is a need for synchronization of time between the different models. In the following we discuss the details:

(a) Managing Complexity of Interoperating Systems. Integration of independently created models can lead to a complex interoperating system of systems that need to be managed efficiently. Understanding the interoperability issues that arise in this context is the main aim of our research. In our proposed approach, we use meta models to describe simulator-related meta-data; a way to infer data transformations, and a means of specifying and automatically executing orchestration. Metamodels can make the underlying simulator more understandable. Additionally metamodels are abstract of lower-level details of integration and interoperability. Other challenges arise from heterogeneity of the data that simulation models need to exchange. Since we are working with existing simulation models, it is necessary to analyze the data types used internally by the simulators. It might be possible to adapt prior work on data transformation and integration [23]. We plan to investigate whether such techniques can be extended to account for or detect potential data exchange issues that will arise.

(b) Correctness. Another challenge in integrated simulation environments is to ensure the correctness of integrated simulation environments (e.g. preserving causality among different simulation models). In particular we focus on time synchronization and data consistency as critical problems that must be addressed to ensure the correct interoperability of the concurrently executing simulators. The simulation clock that controls simulation time during execution of a simulation resides within each simulator itself. Time synchronization mechanisms are needed to ensure causal correctness for models that use different time advancement mechanisms. Most of available synchronization methods need the participants to agree on a common interpretation of time and a common time advancement method. Our goal is to leverage existing simulators, as is, while enabling data interchange between them and to accommodate multiple time management and advancement mechanisms implemented internally in participating simulators, preserving the autonomy of the individual simulators. We need to describe the semantics of the internal time advancement in different simulation models (e.g., whether it is a continuous-time model with observations made at regular time intervals, a discrete-time model with observations only at "ticks," or a discrete-event model with observations only at irregularly spaced event-occurrence times). The spatial coordinate system must also be specified so that different models can be spatially aligned. We came across both issues in our exercise and resolved them with appropriate interpolations of data and transformations to overcome mismatches.

(c) Scalability. Accurate modeling and analysis of large scale and complex scenarios presents a scalability challenge. Modeling such complex scenarios places considerable stress on the system resources. Such scenarios involve lot of entities, e.g. agents with complex behavior operating on dynamic environments. My thesis focuses on interoperability and correctness issues. Scalability is currently another ongoing topic of research in our group [24].

3. PROTOTYPE SYSTEM IMPLEMENTATION AND EVALUATIONS

Our goal is to develop RAISE prototype system and use it to integrate real world simulators to evaluate the validity of our approach. In this section we discuss the implementation issues of RAISE and we present a detailed case study in which we integrate three real world simulators.

Figure 2 shows different modules implemented in our prototype system. The integration of various simulators results in the introduction of certain inter-simulator dependencies, which were not present prior to integration. An example of such a relationship is that smoke from fire simulator can affect someone's health in an activity simulator. Ideally dependencies need to be reflected immediately from one simulator into another simulator; in our framework we define constraints that capture the extent to which

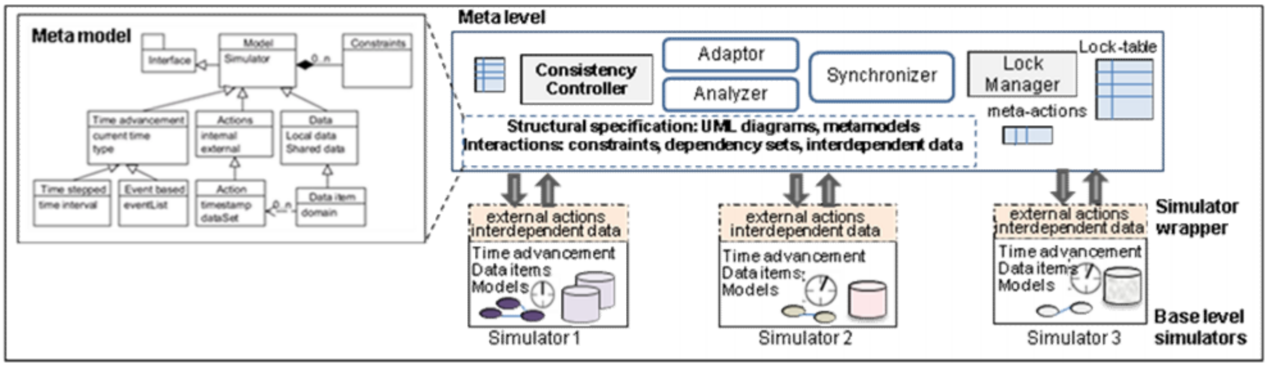


Figure 2. RAISE framework

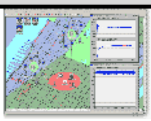
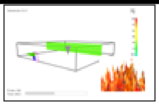
simulators can deviate from ideal behavior. At each iteration a simulator may perform updates on a data item that need to be reflected into another simulator's data item. Metasimulation maintains a queue of meta actions to ensure the dependency between the data items in two different simulators. In order to reduce the synchronization overhead, we use constraints to divide simulators actions (steps in time stepped simulators or events in an event-based simulator) into internal and external actions. Synchronization is only needed during external action processing and we can eliminate synchronization altogether during internal action processing periods.

Meta-synchronizer uses different solutions to monitor and control the concurrent execution of different simulators. Data exchange module provides data transfer that preserves the meaning and relationships of the data exchanged between different simulators (e.g. geometry translators have the responsibility of performing coordinate conversions between geographies that use different coordinate systems). In RAISE, base level simulators consist of simulation applications, which do not merely consist of the simulator code, but also include backend databases and models stored in domain-specific formats which contain aspects of the simulators. Reification of base level simulators' properties yield in metamodels at meta level specified in UML diagrams. The information about simulators dependency sets, constraints and interdependent data items are also represented at meta level. There can be many kinds of metadata entities to cover various integration aspects. The basic metamodel of simulators that contains the information needed in our approach is shown in figure 2. In next section we discuss the detailed case study from the emergency response domain by integrating three disparate simulators to use meta-synchronizer in RAISE.

3.1 Integrating Multiple Real-World Simulators using RAISE

We develop a detailed case study for simulation integration in RAISE using three pre-existing real world simulators – the

Table 1. Three real-world simulators and their properties

Evacuation simulator **Drillsim**	**Communication simulator LTEsim**	**Fire simulator** **CFAST**
 ♦ Time stepped ♦ Open source, agent based (in Java) ♦ Parameters: health profile, visual distance, speed of walking, num. of ongoing calls… Output: num. of evacuees, injuries…	 ♦ Event based ♦ Open source (in Matlab) ♦ Parameters: num. of transmit and receive antennas, uplink delay, network layout, channel model, bandwidth, frequency, receiver noise, etc. Output: pathloss, throughput…	 ♦ Time stepped ♦ Black-box (no access to source) ♦ Parameters: building geometry, materials of construction, fire properties, etc. Output: temperatures, gas concentrations: CO2…

primary goal is to validate RAISE and understand issues in its realization. Our case study is in the domain of emergency response management. The specific simulators in this case study are (1) CFAST, the Consolidated Model of Fire and Smoke Transport, is a fire simulator that simulates the effects of fire and smoke inside a building and calculates the evolving distribution of smoke, fire gases, and temperature [10] and (2) Drillsim, a multi agent evacuation simulator that models a response activity evacuation, e.g. building evacuation in response to an evolving fire hazard [9] and (3) LTESim, a communication simulator for the next generation wireless network infrastructure [15, 16] which abstracts the physical layer and perform network level simulations of 3GPP Long Term Evolution with lower complexity. Table 1 summarizes the three simulators and their properties. In this case study, we focus primarily on integrating simulation and models aimed at informing emergency response policy decision making, but we expect our framework and methods will be applicable to other complex problem domains as well.

In our integration scenario, the fire simulator, CFAST, is used to simulate the impact of fire and smoke in a specific region and calculates the evolving distribution of smoke; fire and smoke can affect evacuation process, e.g. people's health condition, in the evacuation simulator, Drillsim, which has impacts on communication patterns in communication simulator, LTEsim. Such integration is useful to understand various factors that can adversely delay evacuation times or increase exposure and consequently used to make decisions that can improve safety and emergency response time; which enable doing much more better what if analysis.

The first step in using RAISE is to specify metamodels for the three base level simulators and constraints across simulators. We currently envision the specification of metamodels and constraints to be a human-in-the-loop process where the user executing the multisimulation is aware of the aspects being observed and their potential impact in integrated environment. Meta-models are specified as UML diagrams and are invisible to base-level computation. A careful examination of the features in the metamodel and subsequent cause-effect analysis allows users to extract constraints that capture interactions between these simulators. The following are the examples of information interchanged among simulators:

- A harmful condition in CFAST can affect an individual's health in Drillsim.
- Agents in Drillsim can communicate information on the fire and its location – this will prevent agents from entering dangerous areas which might increae the number of ongoing calls (people talk about the crisis) in Drillsim
- Harmful conditions in CFAST can affect the evacuation process in Drillsim e.g. increase walking speed which maps to user speed in LTEsim.
- Smoke in CFAST can decrease an agent's visual distance in Drillsim.
- The number of ongoing communications in Drillsim can affect network pathloss and throughput in LTEsim.
- Pathloss in LTEsim can be used to determine connectivity/coverage in Drillsim.
- Information on building layout from CFAST and Drillsim can determine the number of transmit and receive antenna required in LTEsim.

- Number of evacuees from Drillsim determines the number of users in LTEsim.
- Number of evacuees in Drillsim can affect receiver noise in LTEsim.

Using RAISE we first extract the simulator design information as metamodels from the base-level simulators using their interfaces, source code, and databases. Then using the metamodels capability the model elements that need to be integrated can be extracted. So far we implemented different modules in our framework: (i) the data translators to transform the data between different simulators, (ii) the time synchronizer to synchronize the time between simulators using a range of conservative and optimistic solutions (iii) the geometry transformer to perform coordinate conversion. In near future, we will complete different modules in our framework. We also want to focus on the examination of the support offered by the reflective architecture in integrating simulators in other domains including earthquake and transportation simulators.

3.2 Experimental Evaluations

We will consider different types of performance measures in our evaluations such as timing performance (e.g. the total execution time of simulations, synchronization time overhead), data access and resource consumption performance in data exchange module (e.g. total number of locks, the total number of access conflicts, etc.), scalability performance (e.g. to study the behavior of our approach when number of interactions between simulators increased).

4. CONCLUSION

The goal of presented PhD research is using reflective middleware approach to build a platform that supports the integration of multiple existing models, simulations, and data. First, we have described the motivations for simulation integration along with the related work done in this area and its limitations. Then we discussed some challenges in our research projects. Using our proposed reflective architecture we have obtained a dynamic framework which is flexible, scalable, and easy to implement. To our knowledge this is the first achieved architecture for simulation integration through the use of reflection. Reflection provides the mechanisms needed to access and modify the environment of a given simulator and the flexibility provided by the reflective middleware architecture is a perfect match for the challenges in simulation integration. More specifically we advocate a reflective architecture were each simulator has its own meta level using our suggested metamodel. In near future, we are going to complete our framework and evaluate our proposed techniques via different case studies.

5. REFERENCES

[1] S. M. Metev and V. P. Veiko, Laser Assisted Microtechnology, 2nd ed., Kheir, N.A., Dekker, M: Systems Modeling and Computer Simulation, 2nd ed., Springer, New York, USA (1995)

[2] Pope, A.: The SIMNET Network and Protocols, Technical Report 7102, MA: BBN Systems and Technologies, Cambridge, Massachusetts (1989)

[3] Kon, F., Costa, F., Blair, G., Campbell, R.H.: The Case for Reflective Middleware, Communications of the ACM, 45(6), 33–38 (2002)

[4] Weatherly, R., Seidel, D., Weissman, J.: Aggregate Level Simulation Protocol, Summer Computer Simulation Conference, Baltimore, Maryland, 953-958 (1991)

[5] Ling, Y., Zhang, M., Lu, X., Wang, W., Lao, S.: Model Searching Algorithm Based on Response Order and Access Order in War-Game Simulation Grid, Edutainment, Springer-Verlag Berlin Heidelberg, 627-637 (2006)

[6] Rhalibi, A.E., Merabti, M., Shen, Y.: Improving Game Processing in Multithreading and Multiprocessor Architecture, Edutainment, Springer-Verlag, 669-679 (2006)

[7] Jain, S., McLean, C.R.: Integrated simulation and gaming architecture for incident management training, Simulation, Proc. of the Winter Simulation Conf., 904-913 (2005)

[8] Fujimoto, M.R.: Parallel and Distributed Simulation Systems, John Wiley Inc. (2000)

[9] Balasubramanian, V., Massaguer, D., Mehrotra, S., Venkatasubramanian, N.: DrillSim: A Simulation Framework for Emergency Response Drills, ISI, 237-248, (2006)

[10] Peacock, R., Jones, W., Reneke, P., Forney, G: CFAST–Consolidated Model of Fire Growth and Smoke Transport (Version 6) User's Guide, NIST Special Publication (2005)

[11] Davis, P.K: Distributed Interactive Simulation (DIS) in the Evolution of DoD Warfare Modeling and Simulation, Proceedings of the IEEE 83(8), 1138-1155, (1995)

[12] Mathiassen, L., Reflective Systems Development, Scandinavian Journal of Information Systems, 10(1&2), pp. 67–117 (1998).

[13] HAZUS-MH: Multi-hazard Loss Estimation Methodology. User Manual, (2003)

[14] Cho, S., Huyck, C.K., Ghosh, S. Eguchi, R.T.: Development of a Web-based Transportation Modeling Platform for Emergency Response. 8th Conference on Earthquake Engineering, San Francisco (2006)

[15] Mcqueen, D.: 3GPP LTE: the momentum behind LTE adoption, IEEE communication Magazine, Vol. 47, pp. 44-45 (2009)

[16] LTE System Level Simulator: https://www.nt.tuwien.ac.at/downloads

[17] Jalali, L., Venkatasubramanian, N., Mehrotra, S.: Reflective Middleware Architecture for Simulation Integration, Proc. of the 8th International Workshop on Adaptive and Reflective Middleware, Urbana Champaign, Illinois (2009)

[18] Verkehr, AG: VISSIM Version 3.6 Manual. Innovative Transportation, (2001)

[19] Cooper, L.Y, Forney, G. P.: The consolidated compartment fire model (CCFM) computer code application CCFM.VENTS - Part I: Physical basis. NISTIR 4342. National Institute of Standards and Technology. USA, (1990)

[20] Cameron, G., Wylie, B., McArthur, D.: PARAMICS-Moving Vehicles on the Connection Machine, Conference on High Performance Networking and Computer, conference on Supercomputing, pp. 291– 300 (1994)

[21] CAPARS: http://www.alphatrac.com/PlumeModelingSystem

[22] Abanades, A., Sordo, F., Lafuente, A.: Martinez-Val, J.M., Munoz, J.: Application of computational fluid dynamics (CFD) codes as design tools for inertial confinement fusion reactors, 5th International Conf. on Inertial Fusion Sciences and Applications, (2007)

[23] Haas, L. M., Hernández, M.A., Ho, H., Popa, L., Roth, M.: Clio Grows Up: From Research Prototype to Industrial Tool, Proc. ACM SIGMOD, 805-810 (2005)

[24] Vidhya Balasubramanian, Dmitri V. Kalashnikov, Sharad Mehrotra, and Nalini Venkatasubramanian: Efficient and scalable multi-geography route planning, 13th International Conference on Extending Database Technology (EDBT) Lausanne, Switzerland, (2010)

[25] De Silva, F.N., Eglese, R.W.: Integrating simulation modeling and GIS: Spatial decision support systems for evacuation planning, J. Oper. Res. Soc. 51(4), 423–430 (2000)

[26] Kuhl, F., Weatherly, R., Dahmann, J.: Creating Computer Simulation Systems: An Introduction to the High Level Architecture, New Jersey, Prentice Hall (1999)

[27] Maglio, P. P., Cefkin, M., Haas, P., & Selinger, P.: Social factors in creating an integrated capability for health system modeling and simulation. In *Proceedings SBP 2010.* New York: Springer, pp. 44-51 (2010)

[28] Cefkin, S. M. Glissmann, P. J. Haas, L. Jalali, P. P. Maglio, P. Selinger, W. Tan: Splash: A Progress Report on Building a Platform for a 360 Degree View of Health, M. Proceedings of the 5td INFORMS Workshop on Data Mining and Health Informatics (DM-HI 2010), D. Sundaramoorthi, M. Lavieri, H. Zhao, eds., (2010)