Splash: A Platform for Analysis and Simulation of Health

Wang-Chiew Tan* Peter J. Haas Ronald L. Mak Cheryl A. Kieliszewski
Patricia G. Selinger Paul P. Maglio Susanne Glissmann Melissa Cefkin Yinan Li
IBM Research - Almaden
{wangchiew,phaas,rlmak,cher,patseli,pmaglio,smglissm,mcefkin,liy}@us.ibm.com

ABSTRACT

As asserted by the Institute of Medicine, sound health policy and investment decisions require use of "what if" simulation models to analyze the potential impacts of alternative decisions on health outcomes. The challenge is that high-level health decisions require understanding complex interactions of diverse systems across many disciplines both inside and outside of healthcare, creating a need for experts across widely different domains to combine their data and models. Splash—the Smarter Planet Platform for Analysis and Simulation of Health—is a novel decision support framework that facilitates combining heterogeneous, pre-existing simulation models and data from different domains and disciplines. Splash leverages and extends data integration, search, and scientific-workflow technologies to permit loose coupling of models via data exchange. This approach avoids the need to enforce universal standards for data and models, thereby facilitating both model interoperability and reuse of models and data that were independently created or curated by different individuals or organizations. In this way Splash can help domain experts from different areas collaborate effectively and efficiently to attack complex health problems. We illustrate Splash's architecture and capabilities using a simple, proof-of-concept model of community obesity. We show how models of transportation, eating habits, food-shopping choices, exercise, and human metabolism can be combined with geographic, store location, and population data to play "what if," asking, for instance, how community obesity measures would change if tax incentives are used to encourage grocery chains selling healthy and inexpensive food to open stores near obesity "hot spots."

Categories and Subject Descriptors

D.2.2 [**Design Tools and Techniques**]: Programmer Workbench; D.2.12 [**Interoperability**]: Data mapping; I.6.8 [**Types of Simulation**]: Combined; J.3 [**Life and Medical Sciences**]: Health

General Terms

Design

Keywords

Simulation model composition, schema mappings, workflow, data analysis

*Affiliation: IBM Research - Almaden and UC Santa Cruz. Tan is on leave from UC Santa Cruz.

IHI'12, January 28–30, 2012, Miami, Florida, USA.

1. INTRODUCTION

Making good health policy and investment decisions requires not just the gathering, mining, statistical analysis, and visualization of data, but also the use of "what if" simulation models to analyze the potential impacts of alternative decisions on health outcomes [20, 23]. Such modeling and analysis is very challenging, because high-level health decisions frequently require understanding complex interactions of diverse systems across a great many domains and disciplines, far beyond the healthcare system itself; i.e., genetics, behavior, and environment all interact. For example, the problem of chronic obesity [28] is influenced by a broad array of factors at many levels—such as individual psychology and metabolism, food prices, advertising, transportation, agriculture, education, sanitation, government policies, and international trade—and requires a systems-oriented approach to modeling that brings together experts across disciplines to analyze different aspects of the larger problem [21]. Even the food system considered by itself is complex [17], creating a need for modelers across very different domains to compare models, share results, and integrate efforts. It is not surprising, then, that the Institute of Medicine [23] has recommended that the Department of Health and Human Services advance the use of system-based simulation models to better understand the underlying determinants of health and assess intended and unintended outcomes associated with policy, funding, investment, and resource decisions.

In the area of health informatics, modern technology for information management, data analytics, and computer simulation has been increasingly successful in providing decision support for individual caregivers and healthcare administrators. The key question addressed in this paper is whether such technology can be used to help experts across highly diverse domains collaborate to solve broad, complex health problems. Clearly, no single dataset, model, or knowledge base can capture all factors related to health, so at issue is whether individual well-established models and data from different disciplines can be (incrementally) composed to develop comprehensive models for supporting scientifically sound health decisions. Without any technological assistance, this task is extremely hard, because domain experts have different worldviews, use different vocabularies, sit in different organizations, and have often invested considerable effort in developing and implementing

their models using different programming paradigms and development platforms. Based on our extensive experience in data integration, we feel that any approach to collaboration that requires use of universal data formats or computing environments, or that requires scientists to recode their models to common standards or APIs, is doomed to failure. This paper therefore proposes an alternative approach to facilitating cross-domain collaboration for modeling and simulation of health, and describes a prototype system that implements this approach.

Specifically, this paper presents Splash—the Smarter Planet Platform for Analysis and Simulation of Health—a novel decision support framework that facilitates combination of heterogeneous, preexisting simulation models and data from different domains and disciplines. Splash enables interoperability and reuse of models and data that were independently created or curated by different individuals or organizations, thereby helping domain experts from different areas to collaborate effectively and efficiently to leverage their combined knowledge. The resulting composite simulation models can be used to conduct deep predictive analytics, enabling "what-if" analyses that cut across disciplines, attacking complex health problems that cannot be solved using expertise from only a single domain. We envision that the community of Splash users will include domain experts who contribute models and data, scientists and policy analysts who compose the models and data to run simulation experiments using the composite models, and, ultimately, decision makers who use the results from Splash to inform their policy and investment decisions.

The basic technical approach underlying Splash is semi-automated, loose coupling of models via data exchange. In Splash, models run asynchronously and communicate with each other by reading and writing datasets via file I/O, database accesses, or web-service calls; data transformations are applied as needed for compatability. Fully automated composition of tightly coupled simulation models is well known to be extremely challenging [14]; Splash tries to simplify the problem by exploiting and extending techniques from semi-automated data integration [19]. The idea is to provide graphical user interfaces (GUIs) that simplify both selection of component models and specification of complex mappings between the input and output datasets for these models, while automating more mundane tasks such as conversion of measurement units, spatiotemporal alignment between models, statistical calculations, and so on. Splash's loose-coupling approach minimizes changes that need to be made to existing models, facilitating reuse and collaboration. Splash uses semantic search methods to identify relevant and compatible component models in its repository. Scientific-workflow technologies are exploited to orchestrate execution of the final, composite model; both models and data may be distributed across different platforms. Splash relies heavily on user-supplied metadata about data sources, models, and mappings; this metadata is specified in a unified manner using a novel, XML-based Splash Actor Description Language (SADL).

We describe the architecture and capabilities of Splash in the context of a simple, proof-of-concept composite model of community obesity. In our hypothetical scenario, we show how independently created models of transportation, eating habits, shopping choices, exercise, and metabolism can be combined with geographic data, store location data, and population data to explore the potential consequences of health policy decisions on obesity, as measured by body mass index (BMI). An example of a policy question related to obesity is: "How would a community's average BMI change if grocery chains selling healthy and inexpensive food received tax breaks for building stores near obesity hot spots?" (Such hot spots often correspond to low-income areas.) Our aim in this paper is not to develop a full-scale, validated, policy-ready obesity model. Rather, our goal is to illustrate the basic design and capabilities of our platform to show how our model-composition approach can potentially be used to promote interdisciplinary collaboration around chronic obesity and other complex health issues.

Splash is a work in progress, and so our discussion centers on those features that are already either present or currently being implemented in our research prototype. Throughout, we discuss work that remains to be done, and at the end, we discuss some important open research challenges for Splash.

2. AN OBESITY SCENARIO

In this section, we describe our hypothetical obesity scenario, along with some relevant component models and data sources. In the next section, we describe how Splash can be used to combine these models and data into a composite model suitable for what-if analysis.

In our scenario, government policy makers want to determine the most cost-effective way to reduce obesity—that is, BMI levels—in the population of an urban area. There are many possible approaches. For example, the government could provide incentives for a supermarket chain to place a store that sells healthy and reasonably priced food at a specific location, or build more playgrounds in certain neighborhoods, or give citizens a tax credit for enrolling in an exercise program, or spend more on nutrition education in schools. Which combination of these approaches is most cost effective?

Diverse factors influence obesity levels, including transportation, buying and eating behavior, availability of exercise facilities, and the impact of each on individual and aggregate BMI values. Our obesity scenario entails models of these factors, as well as the data sources that feed these models. The data sources are heterogeneous and independently created, as are the models, which also rely on differing technologies—programming languages, operating systems, simulation paradigms, and so forth—and embody differing assumptions. The component models and data sources used in our proof-of-concept composite model are as follows (see Figure 1).

Buying-and-eating model. We used a simple agent-based model, adapted from [3], that simulates the grocery-store shopping behavior of households over time, tracking each individual household and grocery store in the hypothetical urban area. The model takes into account the food preferences of each household, travel time to the different grocery stores, the household's purchasing history, and social factors, such as where a household's neighbors are currently shopping. Grocery stores may close down as a result of poor sales and new stores may open. The input to the model comprises travel time information (from a transportation model) and a modifiable set of input parameters, such as the percentage of grocery stores serving healthy food and the percentage of high-income households. The model outputs the state of all households and stores at each simulation tick. We implemented this model using NetLogo [33].

Transportation model. We used VISUM [30], a commercial traffic-flow simulation software package that can simulate different modes of public and private transport to determine the impact of traffic demand based on a number of factors, including the road infrastructure and various demand patterns (which may be specified exogenously to VISUM). VISUM outputs various statistics, including the average travel times between different urban zones.

Exercise model. We used a simple discrete-event stochastic simulation model of exercise-facility use that we developed in-house. It takes as input a set of parameters such as number of households,

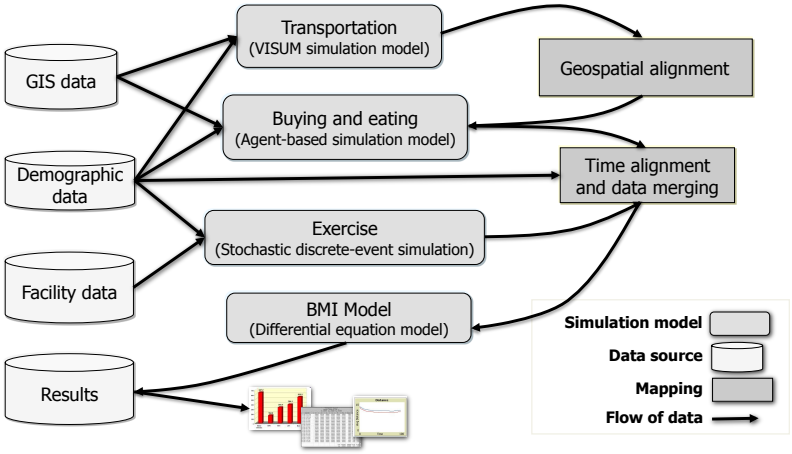


Figure 1: Models, data sources, and mappings for the obesity scenario.

number of exercise facilities, capacity of each facility, base calories burned per unit of exercise time at each facility, and probability distributions that govern the frequency and duration of exercise periods for each household. The model returns as output the number of kilocalories burned by each household member per simulation tick. We implemented this model in C++.

BMI model. We used the "three compartment" energy balance model of Hall and Chow, as described in [29]. This BMI model is a deterministic model that predicts daily BMI changes for an individual based on daily food consumption and physical activity. More specifically, the model uses a set of coupled ordinary differential equations that link exercise level and intake of carbohydrates, fat, protein, and sodium to changes in body mass, broken down as lean mass, fat mass, and extracellular fluid. These body-mass quantities are added and normalized to obtain BMI. Following [29], initial values for various types of body mass as a function of age, gender, height, and weight were obtained from regression models derived from empirical studies. The input to the model is a time series, for each individual, of their daily food intake, as above, and daily kilocalorie burn rate from exercise, and the output is a time series, per individual, of BMI.

Data sources. Our example uses four main data sources. The GIS data source contains geographical information about an urban area, such as road networks and zone configurations. The population-demographics data source contains information about the characteristics of each person in a household, including age, weight, and height. The store-demographics data source contains nutritional characteristics of the food sold at each store. Finally, the exercise-facility data source contains information about the exercise facilities in the hypothetical urban area, such as capacity, type of exercise, and so on. These data sources were synthetically created to test the Splash platform, but publicly available data sources could also be used [11, 31].

3. SPLASH IN DETAIL

The basic architectural components of Splash are illustrated in Figure 2. Using the obesity scenario as an example, we describe the role of each of these components in creating, executing, and analyzing composite simulation models. All components rely on metadata about models, data sources, and mappings—Section 3.2 describes the SADL language used to specify this metadata.

3.1 Using Splash

In Splash, domain experts contribute (and use) component models and data sources. By *component model*, we mean a simulation, optimization, statistical, or other model implemented as a computer program that takes data as input and produces data as output. Each model carries its own set of assumptions and has its own set of requirements and constraints on the input and output data. A *data source* or *dataset* simply refers to a collection of structured or unstructured digital information. Contributors register their models and data sources in the *Splash repository*. A designer of a composite model can then discover these components, connect them together, set up and run simulation experiments, and subsequently analyze, visualize, and share the results; see the center column of Figure 2. The new composite model, as well as any useful datasets generated during the simulation experiments, can be registered in the repository and thus be made available to other model designers.

We now give a detailed description of each Splash workflow step, in the context of our obesity example.

Registration. Models and data must be registered with Splash before they can be used, to create *Splash model actors* and *Splash data actors*. These "actors" are components of Splash that encapsulate the framework's knowledge about the various models and data sources. This knowledge is specified via SADL metadata files that are created by the system as part of the registration process (see Section 3.2 for details). A user can also design and register *Splash mapping actors*, which handle the data transformations between the outputs of one or more models and the input of another. Splash model, mapping, and data actors are connected to each other by the designer of a composite model, and Splash model and mapping actors are invoked for execution during the course of a simulation run.

Model, data, and mapping discovery. Our prototype does not yet support a search capability over the Splash repository, but this functionality will become increasingly important as the size of the repository grows. We envision that to create a composite model in Splash, a user will search through the repository using keyword queries or structured queries over the metadata (SADL) associated with Splash model, data, and mapping actors. Using ontologies and ranking techniques, the discovery component will retrieve and display the "most relevant and compatible" models and data. For example, because the SADL file of the buying-and-eating model mentions "travel times" as one of its inputs, Splash may suggest the

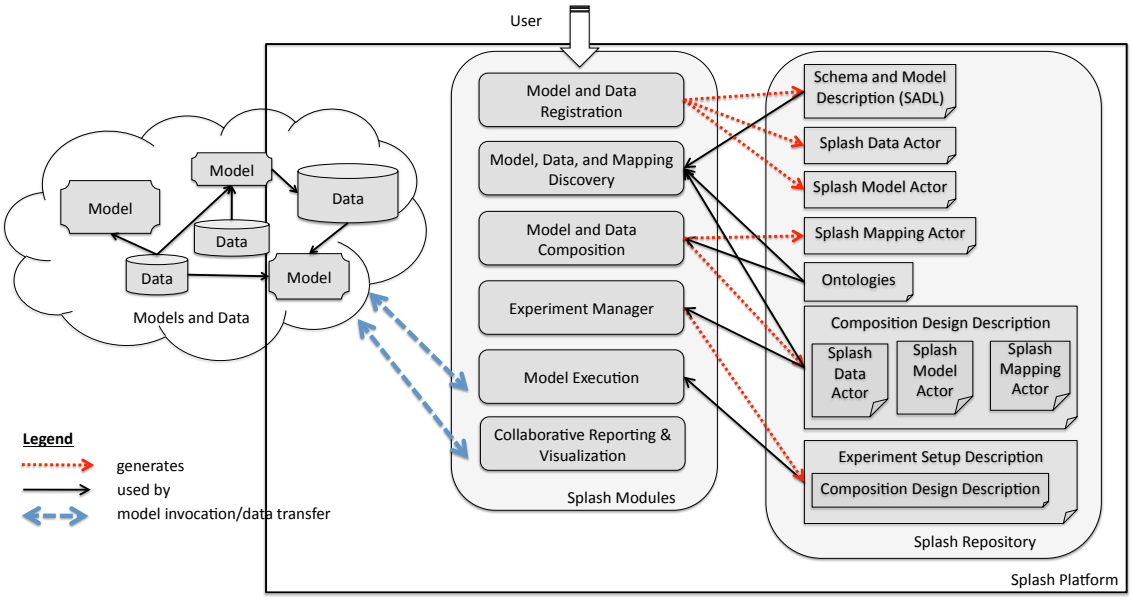


Figure 2: Basic components of Splash.

transportation model as a potentially compatible upstream model because its SADL specifies "travel times" as an output. Ontologies can be especially helpful in dealing with variations of terminology within and between different domains of expertise.

Composite-model design. Splash relies on the Kepler scientific workflow system [25] to provide a visual design environment; indeed, our "actor" terminology derives from Kepler. A user designs a composite model by dragging icons—which represent Splash data actors, model actors, and mapping actors—from the repository window and dropping them into the design workspace. The user then connects the components and configures the mapping actors. Figure 3 displays our composite model for the obesity example. Note that the composite-model structure is immediately apparent from the graphical representation. For example, the join-demographics mapping actor is dependent on the outputs of the buying-and-eating and exercise models, and the buying-and-eating model in turn is dependent on the output of the zone-coordinate mapper. A nice feature of the design environment is that a set of connected Splash actors can be encapsulated as a "complex actor" and then added to the Splash repository; Splash is therefore well suited to the design of hierarchical models.

Because Splash is implemented on an extensible scientific workflow platform, it is relatively easy to add new types of actors to Splash by simply providing appropriate metadata. For example, we could add "data cleaning actors" that could be executed (perhaps during an initialization phase) as part of a composite Splash model. Such actors could deal with situations in which individual data sources are dirty, or in which data sources that provide overlapping information contain inconsistencies. In this way, all data passed to component models will be clean and consistent. Such actors can simply invoke existing software packages for data cleaning and inconsistency resolution.

A key part of designing a composite model is the design of the various data transformations. We next discuss Splash's support for mapping design.

Mapping design. In contrast to tightly-coupled integration frameworks such as [18, 26], models in Splash are loosely-coupled via data exchange; i.e., models are connected via transformations that convert datasets output by one model into a format suitable for input to another model. This loose coupling is possible because every data source is abstracted by a *schema* and every model is abstracted by a pair of input and output schemas. For instance, a schema might specify that patient data is organized as a series of records whose first field is an integer called "patient number," whose second field is a character string called "patient last name", and so forth. Often, the successive records in a data source represent a time series of observations that are input to or output from a simulation model; the schema then specifies the structure of the information recorded at each observation time. *Schema mappings* or *data transformations* refer to specifications of how data is to be translated from one schema (the source schema) into another (the target schema), and are embodied by Splash mapping actors. For example, the "daily protein purchased" attribute in a source schema that corresponds to the output of a buying-and-eating behavior model might be mapped to the "daily protein ingested" attribute in a target schema that corresponds to the input to a human-metabolism model (assuming 100% ingestion of purchased food). To semi-automatically design the different data transformations—that is, to semi-automatically configure the Splash mapping actors—Splash relies on Clio++, an enhanced version of the Clio [19] tool for design of schema mappings.

A fully configured Splash mapping actor represents a *simulation-specific* schema mapping, a declarative description of the transformation from the outputs of one or more models and data (source schemas), to the input of another model (target schema). Such schema mappings are more expressive than traditional schema mappings (as in [7, 19]) because they may contain time-alignment functions that describe how to aggregate, interpolate, or allocate values to overcome mismatches in time interpretations between models, as well as space-alignment functions that describe how to handle geospatial mismatches. A Splash mapping actor contains information about how to execute its data transformation, along with an optional internal representation of the mapping, and can be registered in the Splash repository for later reuse or modification. The process of semi-automating such simulation-specific schema map-

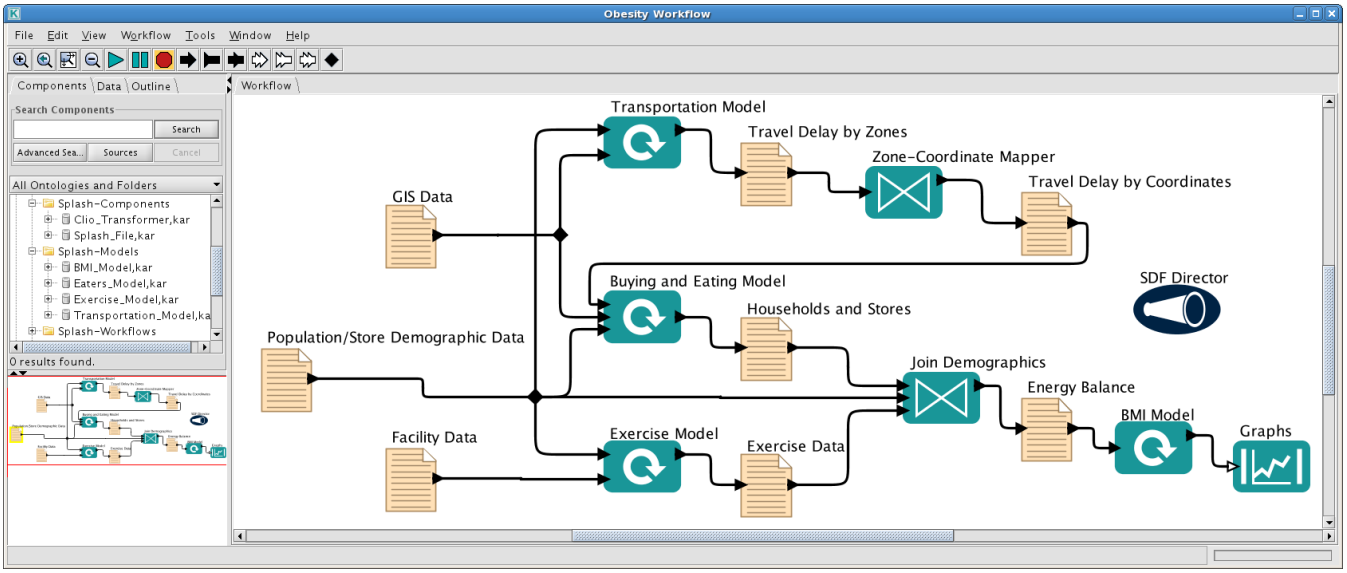


Figure 3: Splash graphical representation of the composite obesity model.

pings relies largely on information derived from SADL files of participating models and data sources.

Our obesity example requires two data transformations:

> **Zone-coordinate mapping.** The transportation model reports average travel times between zones (i.e., regions). However, the buying-and-eating model needs to determine travel times between a household and a grocery store based on geographical coordinates (latitude and longitude). The zone-coordinate mapping is designed to overcome the spatial mismatches between the two models.
>
> **Join-demographics mapping.** This data transformation combines the output of the buying-and-eating model and exercise models with the demographics data into a format that can be used as input by the BMI model. This mapping function also needs to overcome unit mismatches and time mismatches that occur between models (see below).

These data transformations can be specified manually or generated semi-automatically using Clio++. In our example, we manually specified (via a custom Java program) the zone-coordinate mapping, which maps (latitude, longitude) coordinate pairs to the corresponding zones used by the VISUM transportation model. During a simulation run, the Java program is executed on the output data produced by VISUM to create a data file containing travel times between pairs of locations, where each location is specified as a (latitude, longitude) coordinate pair; this file is of the form expected by the buying-and-eating model. We plan to enhance Splash to deal semi-automatically with various kinds of GIS data.

For the join-demographics mapping, we used Clio++ to interactively design the data transformation. During the design process, Splash used the SADL metadata to automatically detect a time mismatch: the buying-and-eating model generates a time series in which a simulation tick corresponds to two days of elapsed time, whereas the BMI model expects a time series in which a simulation tick corresponds to one day of elapsed time. Splash therefore popped up a time-aligner GUI so that the user could specify a desired time-alignment transformation (a linear, cubic spline, or nearest-neighbor interpolation in our example). In general, the GUI will display a menu of suitable time alignment operations, specifying options for aggregating, interpolating, or allocating time series data to effect a desired time alignment; see Figure 4. The time alignment GUI will automatically glean from the SADL metadata the information needed to handle missing data, "boundary" data points at the beginning or end of a time series, and so on.

After the time alignment step, Clio++ popped up a GUI that allowed us to visually specify, for every simulated time step, the relationships between the source schemas (i.e., the demographics data sources and the output files generated by the various models) and the target schema (i.e., the input schema of the BMI model). Figure 5 shows how a user can draw lines to connect attributes in a source and a target schema. This specification determines the procedure by which the demographics data and the various time series produced by the simulation models are joined into the single time series expected by the BMI model. During the process of associating attributes in the source and target schemas, Clio++ automatically generates transformations that correct mismatches in measurement units (such as pounds to kilograms). In the future, Clio++ will also have the capability to draw an initial set of "suggested" lines between the target and source schemas, which can then be tweaked by the mapping designer.

Once the visual specification of a mapping is complete, Clio++ compiles the specification into executable code that is invoked during each simulation run of the composite model. In our current protoytpe, the target platform for such code is Hadoop [2], which runs on commodity hardware and is well suited to robust, massive parallel processing of the large amounts of data created by complex simulation models.

Composite-model execution. For each simulation run of the composite model, Splash uses Kepler's "director" mechanism to orchestrate the execution of the component Splash model actors and mapping actors. In our example, the transportation model executes first, followed by the zone-coordinate mapper. Then the buying-and-eating model may execute in parallel with the exercise model. When both models have completed, the join-demographics mapping is executed, followed by execution of the BMI model.

More generally, a simulation run may consist of multiple Monte Carlo repetitions. Currently, Splash only uses the most basic director functionality, specifying the number of times to execute the

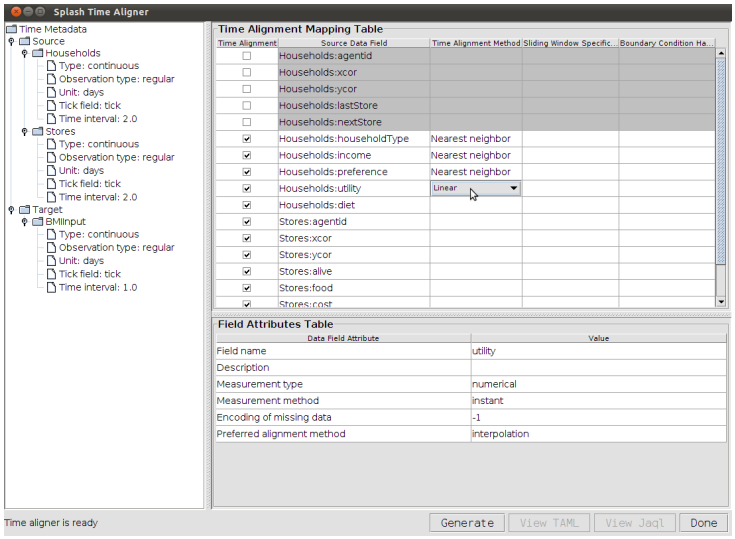


Figure 4: Visual time-alignment interface for designing time alignment transformations.

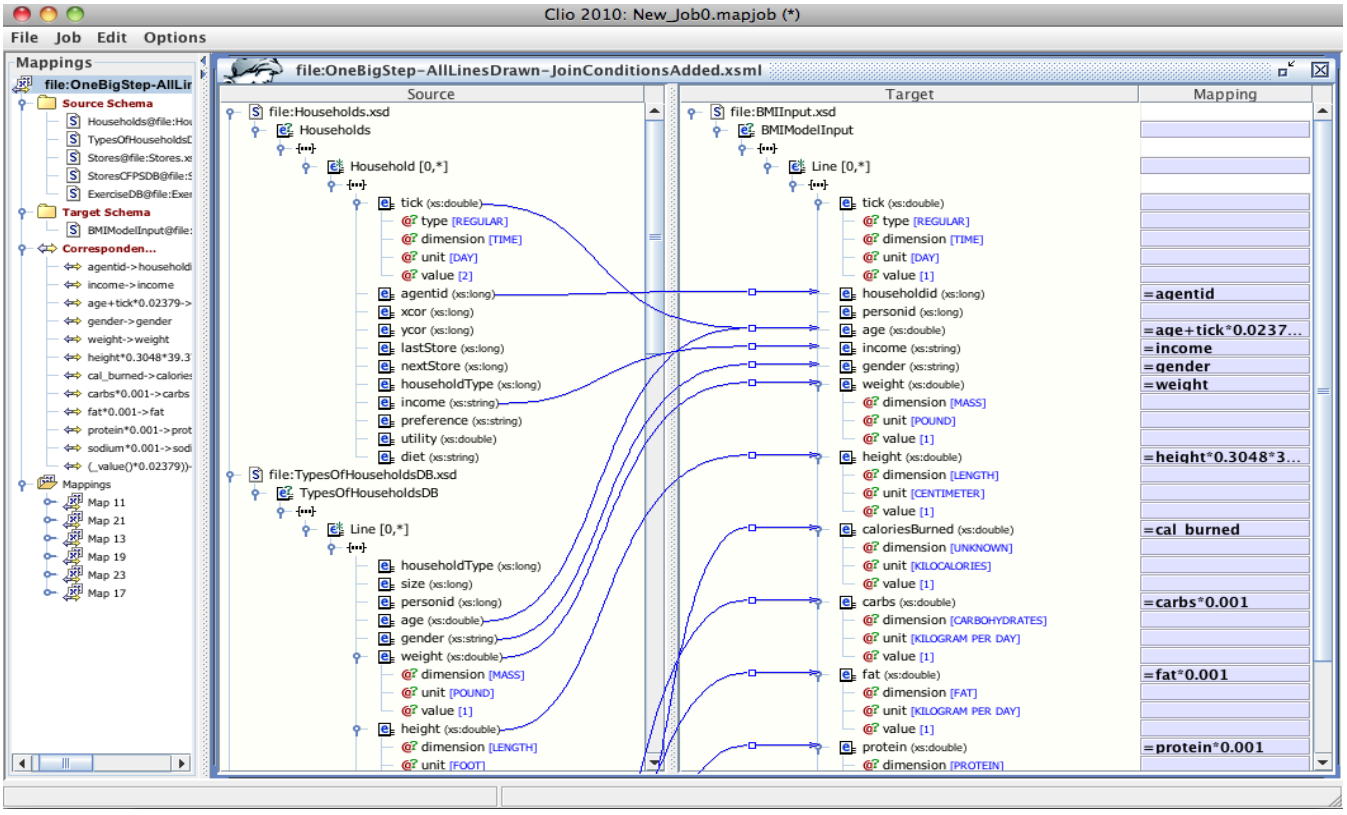


Figure 5: Visual mapping-design for designing structural data transformations.

overall composite model during a run. In the future, we plan to enhance Splash to permit more intricate orchestrations in which the user might specify the number of replications for each component model and how the replications should be combined together. (E.g., for fixed values of the model parameters, a deterministic component model whose input comes only from static data sources should only be executed once per simulation run, since it will produce the same output at every execution.) We also plan to enhance Splash to automatically or semi-automatically deal with statistical issues such as management of pseudo-random numbers [34].

In the current version of our composite obesity model, all models and data sources reside on the same computer as Splash. In general, Splash can execute models remotely; indeed, upon invocation, a Splash model or mapping actor simply synthesizes an appropriate command line string for executing the model or mapping code and sends it to an appropriate destination. This remote execution capa-

bility can be important if certain models must be executed behind a firewall for security or privacy reasons. We intend to enhance Splash with mechanisms for automatically reconfiguring parts of a simulation-experiment workflow among distributed data and models, factoring common operations across different mappings in the workflow, and avoiding redundant computations within a run.

Experiment design. After building a composite simulation model, a user may also design experiments to be run using this model, specifying the model-parameter values to be used for each simulation run in the overall experiment. In the future, we plan to enhance Splash to permit easy specification of experimental designs to support, for example, factor screening and other types of sensitivity analysis for the composite model, as well as simulation-based optimization of model parameters and root-cause analysis of unexpected or unusual simulation results. Such a capability should avoid redundant computations between simulation runs. Another planned enhancement will provide a "dashboard" that gives the user a unified view of all of the parameters for a given simulation experiment or set of experiments; this feature can help users understand the behavior of complex composite models and facilitate interactive experimental design.

Collaborative Reporting and Visualization After running a simulation experiment, the user can review the results. Kepler, and hence Splash, natively supports the R package for statistical analysis, data mining, and visualization [13]. Other packages can be used as well. For example, we visualized the results from sample runs of our composite model by plotting a few simple graphs—see Figure 6—using FusionCharts [16].

Recall that our example is a simple proof-of-concept model on synthetic data, and so the results are merely illustrative and not necessarily realistic. Nonetheless, the graphs illustrate some interesting phenomena. The first graph shows that the average BMI of our hypothetical population decreases over time as a consequence of the opening of a healthy and inexpensive grocery store in a poor neighborhood. The decrease in BMI is more noticeable for the poor population than for the wealthy population, because the latter population already had good access to healthy food prior to the opening of the new store. This simulation assumes that the roads around the store are engineered so that the presence of the new store does not cause additional traffic delays. The second graph shows what happens if the roads are not re-engineered, so that the opening of the new store leads to increased traffic delays. In this scenario, the decrease in BMI becomes much less pronounced: it now takes longer to reach the new grocery store, so fewer people are inclined to shop there and the potential health benefits to the lower-income population are largely unrealized. Thus, even this simple modeling exercise demonstrates the value of combining multiple system models when trying to predict the effect of a proposed health policy decision.

In addition to supporting analysis and visualization tools for individual Splash users, we envision that Splash will also support a collaborative forum similar to ManyEyes [27], in which data and visualizations can be uploaded, shared, annotated, and rated by a community of users.

3.2 Splash Actor Description Language

As can be seen, the metadata specified in SADL files plays a crucial role in Splash, enabling model and data discovery, model composition, and composite-model execution. In this section we illustrate the contents and syntax of the SADL files used by Splash.

Recall that SADL files for models, data, and mappings are created as part of the registration process. For a data source, the provider must specify information such as the schema, data-source location, commands (if needed) to access the data, temporal and/or spatial metadata, and so on. For a model, required information includes the type of model, input and output data sources, and where and how the model is to be accessed and executed. The various schemas are specified in industry-standard XSD format (a dialect of XML).

Figure 7(a) displays a snippet of the SADL file for the BMI model. As can be seen, the description language uses XML-style syntax. The file contains information such as the model's owner—i.e., the user who registered this model in Splash—and references about the model, such as scientific papers, URLs, and reviews. The SADL description also contains information about the history of edits made to the SADL file (not shown), a summary description of the model's functionality, and so on. The `Actor` tag contains basic information about the type of Splash actor being described—model, data, or mapping—using extensible taxonomies of models and data sources. In our example, the SADL description states that the BMI model is a continuous-time, deterministic simulation model.

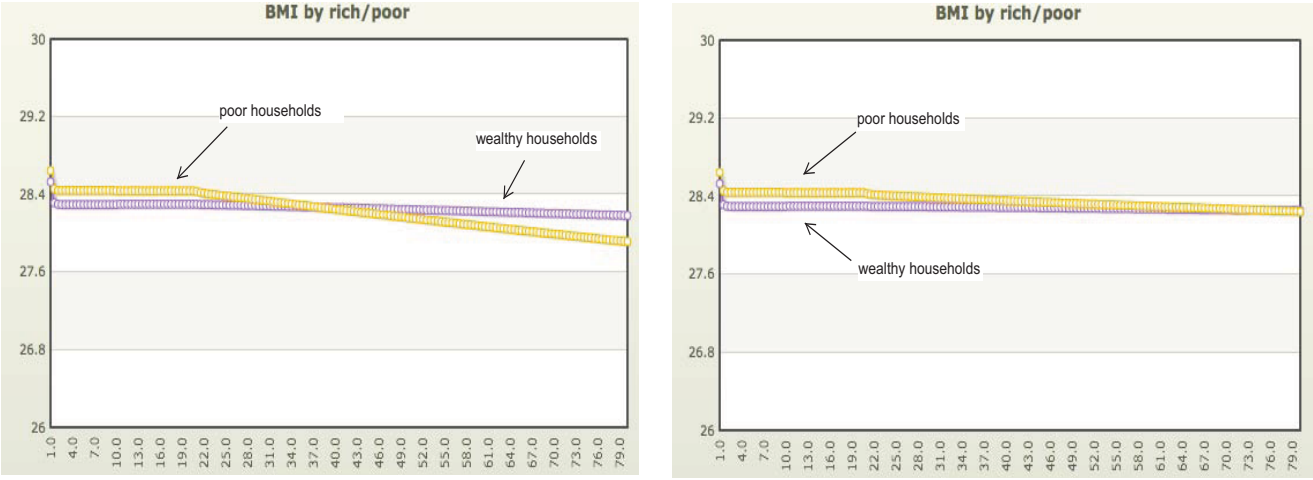
The SADL file also specifies where the model is located and how it is to be executed. In our example, the BMI model resides locally in the directory `$EXEC_DIR/Models`. However, as indicated on the left of Figure 2, not all models and data reside locally to Splash. Some of the models or data may be accessed via web-service calls or other remote execution protocols, in which case the precise invocation method is specified in the SADL file.

Under the `Arguments` tag, the SADL file references two other SADL files—`BMIInput.sadl` and `BMIOutput.sadl`—that describe the data sources corresponding to the inputs and outputs of the BMI model. In general, multiple input and output data sources can be referenced, depending on the data input and output structure of a given model.

SADL file for mappings (not shown here) are similar to those for models, comprising pointers to source and target schema files as well as to the file containing the internal representation of the mapping. Such a SADL file also contains the information needed to invoke the data transformation code during a simulation run.

Figure 7(b) shows a snippet of `BMIInput.sadl`, which describes the input data source expected by the BMI model. In our example, the SADL description states that the data source comprises time-series data in the form of a comma-delimited file. The observations are given at regular intervals, and each tick corresponds to one simulated day of elapsed time. The time appears explicitly in the data file as the `tick` attribute. The path to this file is given by `$EXEC_DIR/Data/BMIInput.del`—thus the file is a local file—and the file conforms to the schema described in `BMIInput.xsd`; a snippet of this latter schema file is shown at the bottom of Figure 7(b). The `BMIInput.sadl` file also describes important characteristics of each attribute (i.e., field) in the data source records, such as measurement units and a description of the semantics of the attribute. For example, `weight` is in pounds, and the associated description states that this measurement is taken before breakfast. Though not shown, the SADL file may also describe general constraints on possible data values, for example, that weight must lie between 0 and 400 pounds or that pre-tax income must exceed after-tax income; such information facilitates both error checking and model composition.

Note that SADL files contain semantic information about models and data at both high and low levels, e.g., in the `description` tags in `BMI.sadl` and `BMIInput.sadl`. Such information is crucial for data and model integration. Semantic mismatches are known to be a major impediment when composing models [14].



(a) **(b)**

Figure 6: (a) Effect on BMI by income of opening a store selling healthy and inexpensive food in a poor neighborhood, and (b) the effect of a traffic increase.

Although dealing automatically with semantic mismatches is very hard, the Splash SADL format encourages detailed semantic information to be maintained for models and data, facilitating semi-automatic avoidance of semantic mismatches. Indeed, the mere process of registering a model or dataset in Splash and generating a SADL file can create valuable documentation. We plan to enhance Clio++ to display low-level semantic information for each schema attribute displayed in the GUIs of Figures 4 and 5.

In addition to the contents described above, a SADL file can include additional information, such as ownership, update history, provenance, and access restrictions. The description language is extensible, and its development ongoing.

4. RELATED WORK

There has been virtually no cross-domain simulation modeling for health. A number of studies on obesity and other health issues have attempted to bridge multiple domains (e.g., [9]), but all of these studies manually bring together a variety of data sources, and then use the data to develop statistical regression or time-series models. Such models can describe existing statistical patterns and corelations well, but can only be used for prediction if the future is—statistically speaking—fundamentally like the past; for example, if historical disease trends continue or if consumers continue to behave as they have up until now. Because statistical models alone do not allow reliable extrapolation to scenarios other than those that produced the historical or training data, such techniques do not provide the deep what-if capability, based on first principles, that serves as the basis for robust decision support. For example, a statistical time series model can reasonably predict rainfall over the next few days, but a simulation model of climate based on the laws of physics is needed to predict the effect of an unexpected increase in CO2 levels on rainfall over the next decade. Splash aims to enhance datasets and statistical models with deep predictive simulation models to provide decision support for health.

The Splash approach to composite modeling—loose coupling via data exchange—was inspired by recent developments in information management, and has not really been explored before, either in health or in other domains. A common alternative modeling approach is to create a "monolithic" model that encompasses all relevant domains. This can be accomplished by writing a single simulation program in a programming language such as C++, Java, or Python, or in a generic simulation programming environment such as Arena [24], which is oriented toward stochastic discrete-event simulation, or AnyLogic [1], which can simultaneously handle discrete-event, agent-based, and system-dynamics models. As the size and scope of such models increases they become extremely difficult and expensive to build, verify, validate, and maintain; see, e.g., [12] or [14, pp. 4–6]. Perhaps more fundamentally, building such a monolithic model typically involves understanding and implementing a body of knowledge in each domain, which limits feasibility in many cases. For instance, consider a monolithic model of the effects of climate on agriculture, and vice versa. Each domain would take years to model and understand, and model development would require continual close interaction between climate and agriculture experts, who generally work in different organizations; such fine-grained collaboration would likely be slow and cumbersome in practice.

For this reason, simulation modelers have been moving toward component models. In an integrated component modeling framework, each component model is a piece of code, usually written in a specified programming language, that is compiled together with other component models into an overall simulation program. Examples of such frameworks include the STEM epidemiological model [15], in which component models are implemented as Java plug-ins to the Eclipse modeling framework, and the Community Climate System Model [10], which comprises atmosphere, ocean, land, and sea-ice component models together with a coupling module, all written in FORTRAN-90. Although an improvement over the monolithic approach, the integrated component approach still requires that the models be written at least to a common interface and often in a specified programming language (another example is OpenMI [18]), which hinders re-use of existing models and can discourage collaboration. As stated earlier, we believe that convincing a broad range of domain experts to program to a common standard is an exceedingly hard task.

Other approaches combine pre-existing models by adding custom logic for synchronized communication across models [26], or requiring models to be written to a specified standard, such as the DEVS framework [32]. Like the monolithic and coupled component approaches, this strategy places heavy burdens on model de-

BMI.sadl

```
<Actor name="BMI Model" type = "model"
       model_type = "simulation"
       sim_type = "continuous-deterministic"
       owner="Jane Modeler">
  <Description>
    Predict weight change over time based on an
    individual's energy intake (food intake),
    sodium intake, and energy expenditure
    (physical activity). Implemented in C.
    reference: http://csel/asu.edu/?q=Weight
  </Description>
  <Environment>
    <Variable name="EXEC_DIR" default="/Splash"
      description="executable directory path"/>
    <Variable name="SADL_DIR" default="/Splash/SADL"
      description="schema directory path"/>
  </Environment>
  <Execution>
    <Command>$EXEC_DIR/Models/BMIcalc.out</Command>
      <Title>Run BMI model</Title>
  </Execution>
  <Arguments>
    <Input name="demographics"
      sadl="$SADL_DIR/BMIInput.sadl"
      description="demographics data"/>
    <Output name="people"
      sadl="$SADL_DIR/BMIOutput.sadl"
      description="people's daily calculated BMI"/>
  </Arguments>
</Actor>
```

(a)

BMIInput.sadl

```
<Actor name="BMIInput" type="data"
  data_type = "time_series">
  <Description>
    Metadata of input data file of the BMI
    model.
  </Description>
  <Time type="continuous" observations="regular"
      field="tick" unit="day" value="1">
  </Time>
  <Data>
    <Source mode="file"
        uri="$EXEC_DIR/Data/BMIInput.del"/>
    <Schema type="xsd"
        uri="$SADL_DIR/Schemas/BMIInput.xsd"/>
    <Format type="del" delimiter=","/>
  </Data>
  <Attributes>
     <name="tick" unit="day"/>
     <name="weight" unit="pound"
       description="weight measured before
                    breakfast"/>
     <name="height" unit="inch"/>
     :
  </Attributes>
</Actor>
```

BMIInput.xsd

```
  :
  <Element name="tick" type="double"/>
  <Element name="weight" type="double"/>
  <Element name="height" type="double"/>
  <Element name="gender" type="string"/>
  <Element name="income" type="string"/>
  :
```

(b)

Figure 7: (a) Snippet of SADL for the BMI model, and (b) SADL for BMI model's input data source and the associated input schema file.

velopers, often requiring deep changes to add the required logic or to modify the model to adhere to the specified standard. Again, these requirements discourage model sharing.

In contrast, Splash is similar to CIShell [8], an open-source software platform that allows different algorithms, datasets, and tools from different domains to inter-operate. The CIShell framework is built upon the Open Services Gateway initiatives (OSGi) framework; algorithms and data sources must be packaged as "OSGi bundles" to inter-operate with other CIShell components. Both CIShell and (future versions of) Splash provide wizard-driven processes to solicit descriptive information (metadata) from the user to generate the OSGi bundles and Splash actors. However, Splash exploits Clio++ to facilitate the design of data mappings that allow previously incompatible models and data to be combined, whereas CIShell does not support semi-automated creation of mappings to translate data from one program to another.

Finally, as is apparent from our description of Splash, we build upon technology both for information integration [6, 19] and for scientific workflow processing [4, 5, 22, 25]. Splash is well positioned to exploit future technological advances in these areas.

5. CONCLUSIONS AND FUTURE WORK

We have outlined the capabilities and design of the Splash platform for combining heterogeneous existing models and data to support complex health decisions. Splash enables cross-domain "what if" analyses that can help avoid unintended consequences and identify truly effective solutions to health issues. Our discussion has been in the context of an obesity model, but clearly our technology is potentially applicable to a broad range of applications in healthcare, public health, and beyond.

As suggested, there is much work to be done in enhancing the Splash prototype with respect to ontology-aided model search, geospatial alignment tools, advanced experimental design capabilities, efficient simulation execution in the presence of distributed models and data, enforcement of statistically valid Monte Carlo techniques, comprehensive visualization of model inputs and outputs, and technologies for deep collaborative modeling and analysis. Moreover, it is clear that issues of data privacy and security are critical in the context of health decisions, and so must be handled effectively by Splash. Another fundamental challenge is dealing with bidirectional causality between models. For instance, it may be feasible to approximate such causality by running models independently but periodically exchanging data. Providing (and justifying) such functionality poses both theoretical and system-design challenges.

Beyond these considerations, there is a range of broader questions to explore. What can be said at a general level about model compatibility and patterns of model interaction? How can we most effectively develop an active ecosystem of model users and model providers around Splash? How can we decide when we have integrated a sufficient number of component models and data? How can we validate the correctness of a composite model and explain the results obtained? Can we leverage available validation results

for component models? How can the output from a Splash-based analysis be turned into actionable decisions or policies?

A key step in addressing the issues raised above is to build "industrial strength" composite models, bringing state-of-the-art component models and comprehensive, high quality data to bear on key health issues. The time is right for harnessing decades of advances in computer engineering, information management, optimization, statistics, and simulation technology in the service of better decision making for health. Splash represents an initial step along this path.

Acknowledgements

We thank Thomas Friderich and Robert Shull of PTV AG for providing the VISUM traffic model, Daniel Rivera for providing the BMI model, Bertram Ludäscher and colleagues at UC Davis for help with Kepler, and Leila Jalali for help in developing the first version of the composite obesity model.

6. REFERENCES

[1] Anylogic. http://www.xjtek.com.

[2] Apache Hadoop. https://hadoop.apache.org.

[3] A. Auchincloss, R. Riolo, D. Brown, J. Cook, and A. D. Roux. An agent-based model of income inequalities in diet in the context of residential segregation. *Amer. J. Preventive Medicine*, 40(3):303–311, 2011.

[4] R. Barga, J. Jackson, N. Araujo, D. Guo, N. Gautam, and Y. Simmhan. The Trident scientific workflow workbench. *IEEE Intl. Conf. on eScience*, pages 317–318, 2008.

[5] L. Bavoil, S. Callahan, P. Crossno, J. Freire, C. Scheidegger, C. Silva, and H. Vo. Vistrails: Enabling interactive multiple-view visualizations. In *IEEE Visualization 2005*, pages 135–142.

[6] P. A. Bernstein and L. M. Haas. Information integration in the enterprise. *Commun. ACM*, 51(9):72–79, 2008.

[7] A. Bonifati, E. Q. Chang, T. Ho, L. V. S. Lakshmanan, and R. Pottinger. Heptox: Marrying XML and heterogeneity in your P2P databases. In *VLDB*, pages 1267–1270, 2005.

[8] K. Börner. Plug-and-play macroscopes. *Commun. ACM*, 54(3):60–69, 2011.

[9] F. J. Chaloupka and L. M. Powell. Price, availability, and youth obesity: evidence from Bridging the Gap. *Prev. Chronic Dis*, 6(3), 2009.

[10] W. D. Collins, C. M. Bitz, M. L. Blackmon, G. B. Bonan, C. S. Bretherton, J. A. Carton, P. Chang, S. C. Doney, J. J. Hack, T. B. Henderson, J. T. Kiehl, W. G. Large, D. S. Mckenna, B. D. Santer, and R. D. Smith. The community climate system model version 3 (CCSM3). *J. Climate*, 19:2122–2143, 2006.

[11] Community Health Data Initiative. http://www.cdc.gov/nchs/data_access/chdi.htm.

[12] R. W. Conway and J. O. McClain. The conduct of an effective simulation study. *INFORMS Trans. Education*, 3(3):13–22, 2003.

[13] P. Dalgaard. *Introductory Statistics with R*. Springer-Verlag, 2002.

[14] P. K. Davis and R. H. Anderson. *Improving the Composability of Department of Defense Models and Simulations*. RAND Corporation, Santa Monica, CA, 2003.

[15] D. A. Ford, J. H. Kaufman, and I. Eiron. An extensible spatial and temporal epidemiological modelling system. *Int. J. Health Geographics*, 5(4), 2006.

[16] Fusioncharts. www.fusioncharts.com.

[17] H. Godfray, J. Pretty, S. Thomas, E. Warham, and J. Beddington. Linking policy on climate and food. *Science*, 331(6020):1013–1014, 2011.

[18] J. Gregersen, P. Gijsbers, and S. Westen. OpenMI: Open modelling interface. *Journal of Hydroinformatics*, 9(3):175–191, 2007. http://www.openmi.org.

[19] L. M. Haas, M. A. Hernández, H. Ho, L. Popa, and M. Roth. Clio grows up: From research prototype to industrial tool. In *SIGMOD Conference*, pages 805–810, 2005.

[20] P. J. Haas, P. P. Maglio, P. Selinger, and W.-C. Tan. Data is dead... without what-if models. In *Proc. of the VLDB Endowment*, 2011.

[21] T. T. Huang, A. Drewnowski, S. K. Kumanyika, and T. A. Glass. A systems-oriented multilevel framework for addressing obesity in the 21st century. *Preventing Chronic Disease*, 6(3), 2009.

[22] D. Hull, K. Wolstencroft, R. Stevens, C. Goble, M. Pocock, P. Li, and T. Oinn. Taverna: a tool for building and running workflows of services. *Nucleic Acids Res.*, 34:729–732, 2006.

[23] Institute of Medicine. *For the Public's Health: The Role of Measurement in Action and Accountability*. The National Academies Press, 2010.

[24] W. D. Kelton, R. P. Sadowski, and N. B. Swets. *Simulation with Arena*. McGraw-Hill, fifth edition, 2010.

[25] Kepler Scientific Workflow System. http://kepler-project.org/.

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

[27] Many Eyes. http://www-958.ibm.com/software/data/cognos/manyeyes/.

[28] A. H. Mokdad, M. K. Serdula, W. H. Dietz, B. A. Bowman, J. S. Marks, and J. P. Koplan. The spread of the obesity epidemic in the united states, 1991–1998. *J. Amer. Medical Assoc.*, 282(16):1519–1522, 1999.

[29] J. Navarro-Barrientos, D. Rivera, and L. Collins. A dynamical systems model for understanding behavioral interventions for weight loss. In *Proc. of Int'l Conf. Social Computing, Behavioral Modeling, and Prediction (SBP)*, pages 170–179, 2010.

[30] Planung Transport Verkehr AG. VISUM traffic simulation model. http://www.ptvag.com/software/transportation-planning-traffic-engineering/software-system-solutions/visum/.

[31] United States Census Bureau. http://www.census.gov/.

[32] G. A. Wainer. *Discrete-Event Modeling and Simulation: A Practitioner's Approach*. CRC Press, 2009.

[33] U. Wilensky. *NetLogo*. Center for Connected Learning and Computer-Based Modeling, Northwestern University, 1999. http://ccl.northwestern.edu/netlogo/.

[34] F. Xu, K. Beyer, V. Ercegovac, P. J. Haas, and E. J. Shekita. $E = MC^3$: Managing uncertain enterprise data in a cluster-computing environment. In *ACM SIGMOD*, pages 441–454, 2009.