

## **Integrated Simulation and Design Synthesis**

David Wallace, Elaine Yang, Nicola Senin

MIT room 3-455  
77 Massachusetts Avenue  
Cambridge, MA, 02139  
drwallac@mit.edu

### **Abstract**

The potential benefits of mathematically predicting and analyzing the integrated behavior of product concepts throughout the design synthesis cycle are widely recognized. Better up-front integrated design will not only reduce development time and cost, but also will yield higher quality products with improved performance. Many academic researchers and companies have attempted to develop integrated simulation environments, and it has been observed consistently that significant difficulties arise because of the large scale, complexity, rate-of-change, heterogeneity, and proprietary barriers associated with product design synthesis. However, the focus of most integration efforts has been on enabling technology, while the process of how integrated systems are constructed has not been questioned.

The literature acknowledges that it is very difficult to represent and structure emergent processes using explicit system definition techniques like those that have been almost universally adopted. The belief that design synthesis is an emergent system definition process drives the search for a different approach to building integrated design simulations. Inspired by a vision of the World-Wide Web as an emergent information-network building environment, a World-Wide Simulation Web concept is proposed for defining an emergent, integrated, simulation-building environment. Participants should be able to make interfaces to local sub-system simulations parametrically operable and accessible over the Internet. Furthermore, any participant should be able to make relationships between parameters in different simulation interfaces or to create additional models that bridge interfaces to different simulations distributed over the Internet.

The DOME (Distributed Object-based Modeling Environment) project has developed a software infrastructure for the purpose of refining and testing emergent simulation definition concepts. A federating solving mechanism has been developed that allows local solvers to respond in a manner that is consistent with the overall system structure even though there is no centralized coordination of the simulation. Results from several pilot studies support the belief that an emergent, decentralized approach to building integrated simulations can resolve many of the difficulties associated with integrated system simulation.

## Motivation

The potential benefits of mathematically predicting and analyzing the integrated behavior of product concepts throughout the design synthesis cycle are widely recognized.

Integrated simulations allow a multitude of what-if scenarios and iterations to be explored inexpensively and quickly.

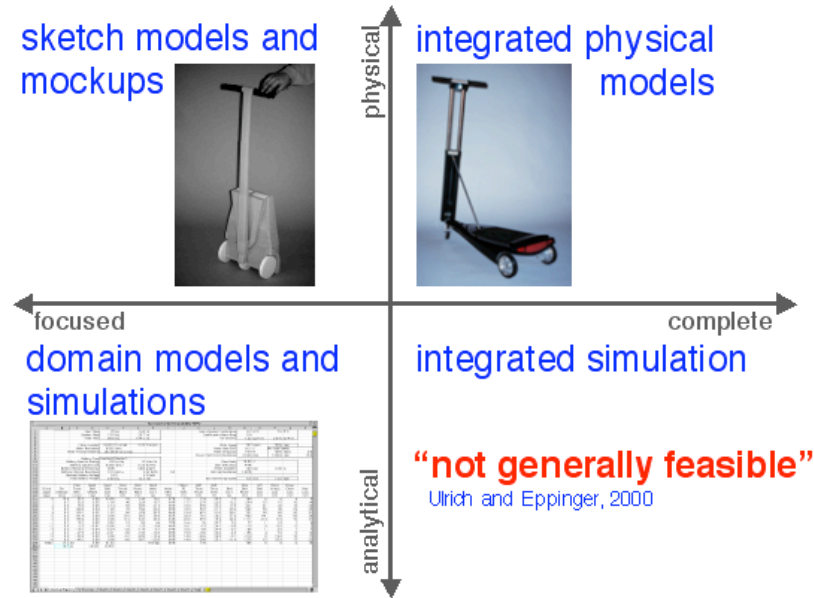
It is estimated that 30% or more of a design engineer's time is spent in meetings to gather or exchange information about interrelated aspects of a product (Christian *et al.*, 1996).

Researchers at Ford Motor Company estimate they spend hundreds of million dollars per year on integration rework (Sferro, 2001). In contrast, studies at Ford (Abrahamson *et al.*, 2000) and at Polaroid (Abrahamson *et al.*, 1999) show that design assessments requiring weeks or months in a traditional design environment can be understood in seconds using integrated simulations, and a LG Electronics pilot study suggests that they will reduce overall concept-to-first prototype development time of air conditioners by 50 percent (Pahng, 2000).

In addition to reducing development time and cost, up-front integrated design will yield higher quality products with better performance. If the implications of changes can be better understood through integrated simulations, designers might be liberated to try more innovative solutions without the fear of unforeseen consequences.

Since the many potential benefits of integrated simulation are generally accepted, there have been many efforts to develop integrated modeling environments in both academia (e.g., Wong and Sriram, 1993; Toye and Cutkosky, 1994; Wellman, 1994; Molina and Al-Ashaab, 1995; Bliznakov and Shah, 1996; Case and Lu, 1996; Cutkosky and Engelmores, 1996; Dabke and Cox, 1998; Kim and Kim, 1998; Wu *et al.*, 2001) and in industry (e.g., Engineous Software, 2001; Fiper, 2001; Phoenix Integration, 2001; RDCS, 2001; Slate, 2001). The different systems employ a wide variety of architectures, data models and communication technologies, but all rely on some form of a consolidated explicit description of the complete integrated system model.

Figure 1 summarizes the different categories of modeling tools that are used during the product design cycle. The vertical axis is divided into physical or analytical models, and the horizontal axis ranges from focused domain-specific models to comprehensive or integrated models. Overlaid upon these axes are examples of models typically used in design practice. The specific examples are from an electric scooter development project in which one of the authors participated.



**Figure 1** Types of models used in product design and development (based upon Ulrich and Eppinger, 2000).

Within the focused/analytical quadrant of Figure 1 numerous simulation tools, ranging from spreadsheets to CAD models and sophisticated CAE analyses, have been applied to product design successfully. However, the comprehensive/analytical quadrant in which integrated simulation resides is deemed not generally feasible (Ulrich and Eppinger, 2000) even though there have been many efforts to develop integrated simulation environments. Given the widespread existence of focused analytical tools, the problem appears to reside in transforming individual analytical models and simulations into an integrated whole.

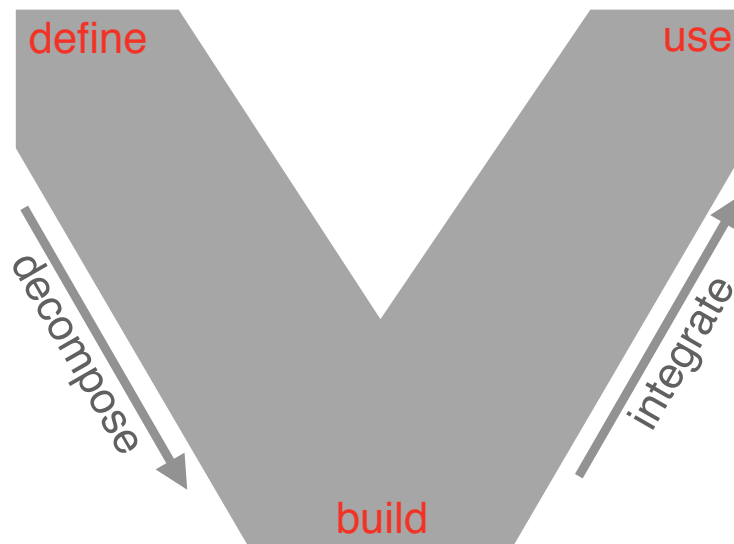
Cooper and Kaplan note that numerous integration efforts within firms over the past several years have had limited success due to the difficulty of creating an explicit model for a very large system involving many suppliers, a rapidly changing product, and an evolving organizational environment (Cooper and Kaplan, 1998). Similarly, other researchers have noted that significant barriers to building integrated product design simulations arise because of the scale, complexity, rate-of-change, heterogeneity, and proprietary barriers associated with comprehensive product design synthesis (Cutkosky, *et al.*, 1994). In fact, since the system-level interactions between participants and tools are often rigidly codified and difficult to change, integrated systems can have negative effects by freezing the system-level design of a product (Wellman, 1994).

**Hypothesis**

An alternative hypothesis about why large integration efforts have had limited success is that the difficulty in building integrated simulations for use during design synthesis arises because the methods used for structuring and building integrated simulations do not support the nature of design activities and the product evolution process. There is a

mismatch between the nature of large-scale design synthesis activities and current integrated simulation model building techniques.

Figure 2 is a stylized schematic of the widely adopted system engineering ‘V’ process for architecting systems. The ‘V’ approach and other similar methods patterned after IDEF (Integration DEfinition for Function Modeling) are widely used in industry for structuring integrated design simulation environments.



**Figure 2** Schematic representation of a system engineering ‘V’.

Starting from the top left of the V, the system is first fully defined or scoped through a top-down, cascading decomposition of subsystem requirements. Then, individual subsystems are constructed, individually tested, and integrated by working back up the V. When one reaches the top right of the V, the system is complete and ready for deployment or use.

This is an explicit, procedural process for defining and building systems. The method assumes that the definition of the overall system and control of its execution is consolidated. The system is usable when it is deployed in a complete form and as a result the architecture of the system is fixed. Designers can set the values of parameters but not change the ‘plumbing’ of the integrated system simulation.

The authors have observed that explicit procedural system definition methods can be readily applied to design or simulation synthesis problems that are addressed by a single individual or small number of individuals, but for larger products, even ones that are quite stable and mature, organizations struggle to define and order their functional system model explicitly. Furthermore, the resulting system definition often seems to have little relevance to the activities and interactions of designers creating the product. For example, even though Ford Motor Company has a structured system requirements-oriented product development process, over 70% of system knowledge has not been captured explicitly.

These uncaptured aspects of the product system reside as fragments within the minds of the domain experts (Dong, 1999).

Observations of design practice in industry and personal experience suggest that design processes involve rapid synthesize/test/evaluate cycles. This may occur in a bottom up, top down, or middle out fashion. During these cycles many discoveries are made, and the design or system undergoes many fluid transformations. For example, David Laws describes a case in California where, through an organic discovery process, an initial design problem to developing an alternative fuel bus system was transformed into a maintenance program design problem (Laws, 2000). Finally, many participants interact and make local decisions during the design synthesis process, and the collective result defines a complex whole.

These observations about the design synthesis process are all consistent with the properties of emergent systems (Wang, 2000). Most complex systems occurring in nature or involving human behavior are emergent in nature. The economy is often cited as an example of an emergent system. Both over-managed and under-managed product development processes result in lengthy design lead time and high development cost (Ahmadi and Wang, 1999). Effective organizations can arise through an emergent process within appropriate environments where individuals can understand the implications of their actions on overarching goals (Hameri and Nihtila, 1997; Coleman, 1999).

It is also recognized that managing the development of emergent systems through strict, explicit, consolidated approaches is very inefficient (Truex *et al.*, 1999). Thus, if the process of designing complex products is an emergent human activity, this may explain the difficulties that have been encountered in the past when building integrated design simulations to support synthesis.

Therefore, the goal of this paper is to describe a new integrated simulation building technology that will allow integrated system simulations to emerge throughout the design activity. It is proposed that the traditional difficulties associated with scale, complexity, rate of change, heterogeneity, and proprietary information can be resolved through an emergent simulation synthesis process that, without consolidated control, can maintain parametric consistency between the distributed simulation models.

### **Technology Inspiration**

There are numerous examples of natural emergent systems, but computer-based structures that develop in an emergent fashion are of particular relevance to this work. The World-Wide Web (WWW), which is a technological embodiment of the hypertext concept originally proposed by Vannevar Bush (Bush, 1945), is an excellent example of such a system.

The WWW has revolutionized the process for building complex networks of information. Any individual with Internet access can easily discover the materials of others and

conveniently add content (i.e., create web pages). These properties of the WWW allow participants to interact freely with the ideas of many other participants.

Furthermore, individual participants can perform system synthesis building upon the work of others by creating links between pages. This ad-hoc, decentralized definition of relationships allows for fluid and continuous transformation of the WWW topology. As a result of these characteristics, the WWW enables an emergent process for building information networks.

### **An Analogous Concept: A World-Wide Simulation Web**

A World-Wide Simulation Web (WWSW) with characteristics similar to the WWW may allow large integrated simulations to form in an emergent fashion.

In a WWSW, any participant will be able to define and publish parametrically operable interfaces to their sub-system simulations on the Internet—similar to how html pages are published in the WWW.

In a WWSW, any participant will be able to define mathematical links, or relationships, between parameters in different simulation interfaces to create additional models that bridge the interface parameters of other simulations. Participants will define these relationships in a declarative fashion that does not require an understanding of the global structure of the integrated simulation or the sequence in which relationships should be executed. The emergent network would be responsible for solving the decentralized relationships and maintaining the mathematical consistency of inter-related simulations.

Enabling individuals to locally declare relationships between simulation interface parameters available via the Internet will result in distributed integrated simulations that emerge and evolve rapidly. If the WWSW embeds appropriate decentralized solving mechanisms, difficulties associated with large scale, complexity, and rate of change when using explicit system definition processes may be resolved.

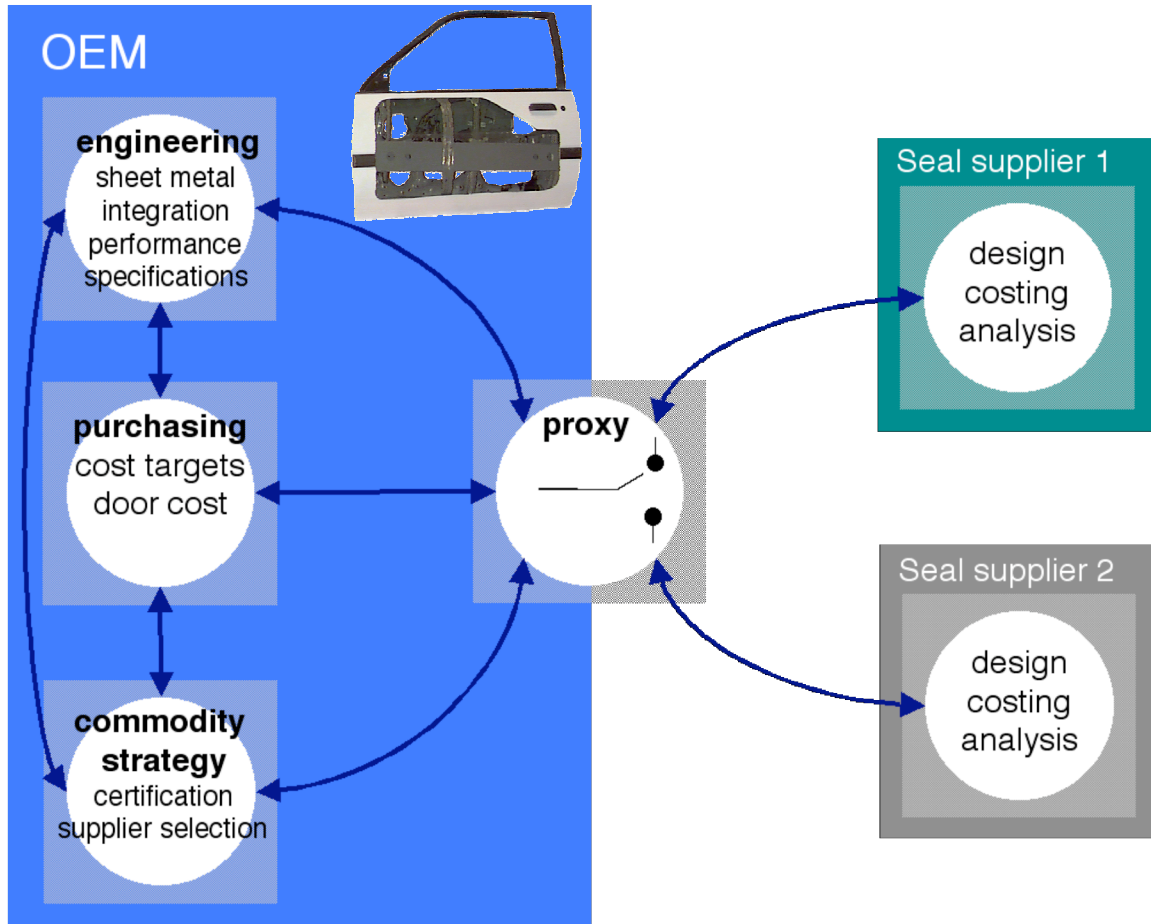
### **Synthesis in a WWSW: An Application Example**

The DOME (Distributed Object-based Modeling Environment) project is an ongoing effort to develop and test a software infrastructure for the WWSW concept. Just as the original hypertext concept evolved to its current WWW embodiment through several generations of implementation, the DOME project is now in the early stages of its third generation implementation. The first-generation implementation was developed in 1996 and emphasized computation, decision support, and optimization (Pahng *et al.*, 1998), whereas the second-generation implementation focused on simulation marketplace concepts using CORBA as a distributed communication protocol (Abrahamson *et al.*, 2000; Senin *et al.*, 2000).

The new third-generation implementation used for the example in this section focuses on combining http protocol-based simulation service marketplace concepts with the efficient solving of emergent integrated simulations. The process of building an emergent system simulation—which is a key element of the WWSW concept—is illustrated through a

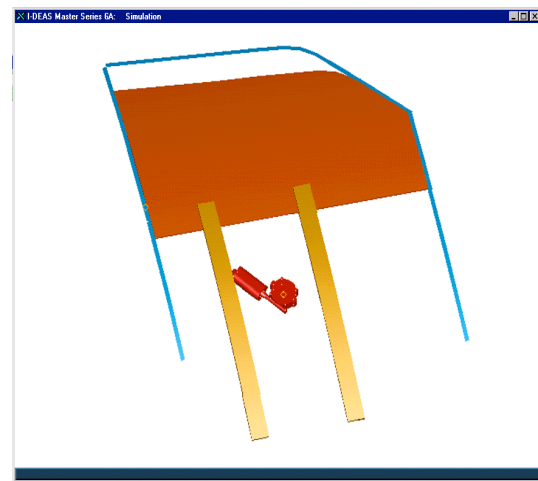
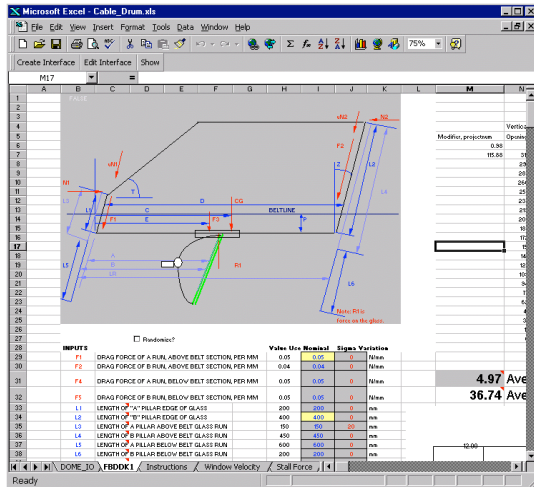
simple scenario that draws upon elements from a second-generation DOME pilot study with Ford Motor Company that was conducted using the second-generation DOME implementation (Abrahamson *et al.*, 2000).

Figure 3 illustrates the scope of the full door glass drop study conducted at Ford. Models in three different internal organizations and two external suppliers were integrated to form an integrated glass-drop sub-system simulation. The integrated simulation involved 21 models in different 3<sup>rd</sup> party applications and the coordination of some 1500 shared design parameters.



**Figure 3** Scope of the Ford glass-drop study.

One simulation model from the glass-drop application is a spreadsheet developed by a Ford glass-velocity engineer. The spreadsheet is used to predict stall force and velocity characteristics of a door's window glass. A second model is a geometric representation used by a CAD designer to maintain the glass-drop geometric configuration. These models are shown in Figures 4a and b.

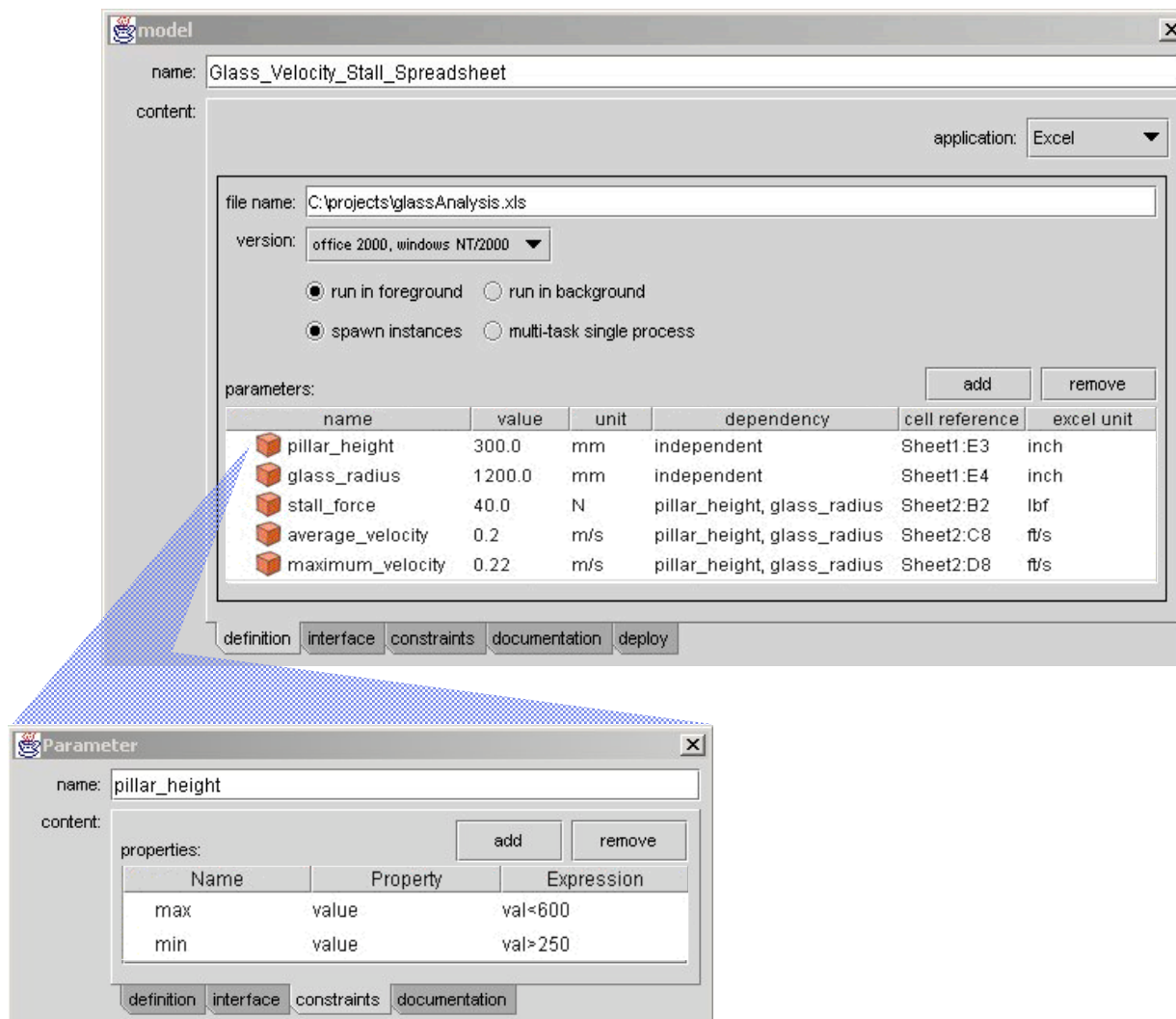


**Figure 4a** Stall force and velocity spreadsheet. **Figure 4b** CAD representation of the glass-drop.

In the first step of building an integrated simulation, model owners *build* and *deploy* parametric interfaces to their simulations so that other participants can interact with their simulations over the Internet.

Figure 5 illustrates this process as executed by the glass-velocity engineer. Using a DOME building environment for wrapping third party applications, the glass-velocity engineer has created DOME parameter objects named pillar\_height, glass\_radius, stall\_force, average\_velocity, and maximum\_velocity.





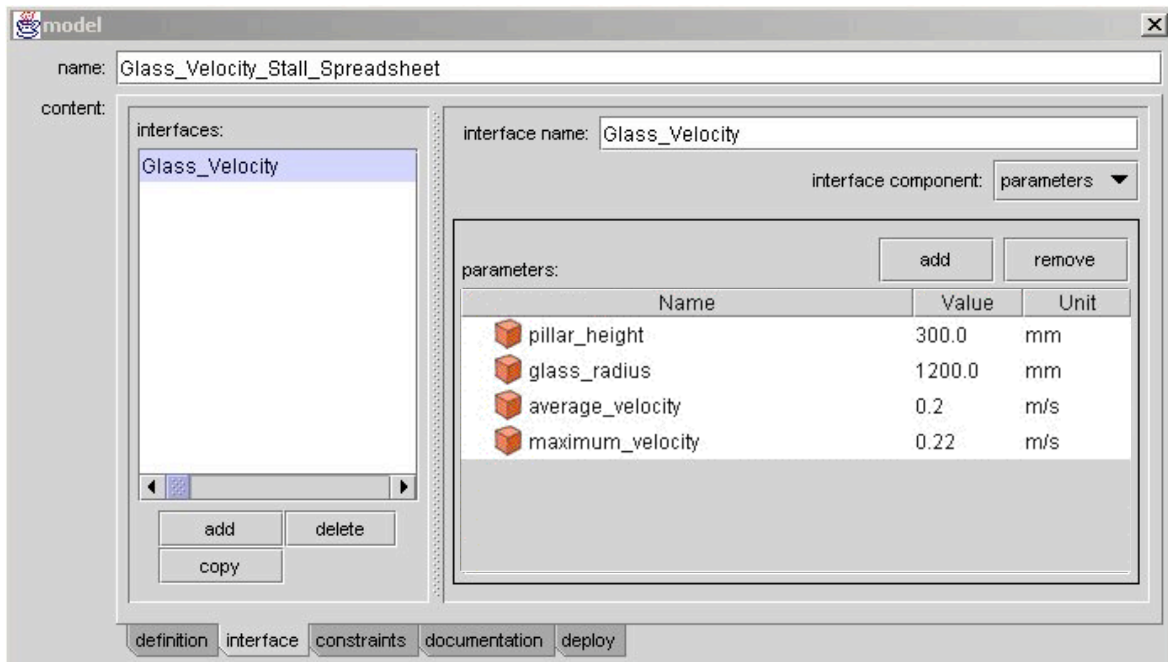
**Figure 5** Defining a DOME wrapper model for the stall force and velocity spreadsheet simulation.

In Figure 5 the glass-velocity engineer has defined unit and value constraints on the DOME parameters. For example, the engineer wants to limit pillar\_height values to range between 250 and 600 mm. DOME parameters can assume the form of several data types (e.g., real number, integer, vector, matrix, boolean, string, list, table, or file), but in this case the DOME parameters are real number objects because they are mapped to numeric cells within the spreadsheet. The glass-velocity engineer has also specified default values for the DOME parameters. The default values are used to initialize the underlying spreadsheet simulation and to provide placeholder values in case the spreadsheet simulation is not operable. The internal units assumed by the glass-velocity engineer when building the spreadsheet simulation are also specified so that DOME can convert parameter values, if necessary, before sending values to the spreadsheet.

Finally, the glass-velocity engineer provides a causal map that described how the DOME parameters are related to each other through the spreadsheet simulation model. In this case, pillar\_height and glass\_radius are independent parameters, while stall\_force, average\_velocity, and maximum\_velocity are dependent upon both pillar\_height and

glass\_radius. The glass-velocity engineer can define the causal mapping manually, or could use a tool that systematically executes the simulation to infer causal relationships.

Once the glass-velocity engineer has defined the contents of the DOME wrapper model, the next step is to define interfaces to the DOME model. An interface determines what subset of model parameters will be exposed to subscribers of the simulation. It is possible to define many interfaces to the same model so that different subscribers will have access to different parameter sets. In this example (Figure 6) a single interface called Glass\_Velocity has been defined, and the engineer has added the DOME parameters pillar\_height, glass\_radius, average\_velocity, and maximum\_velocity.

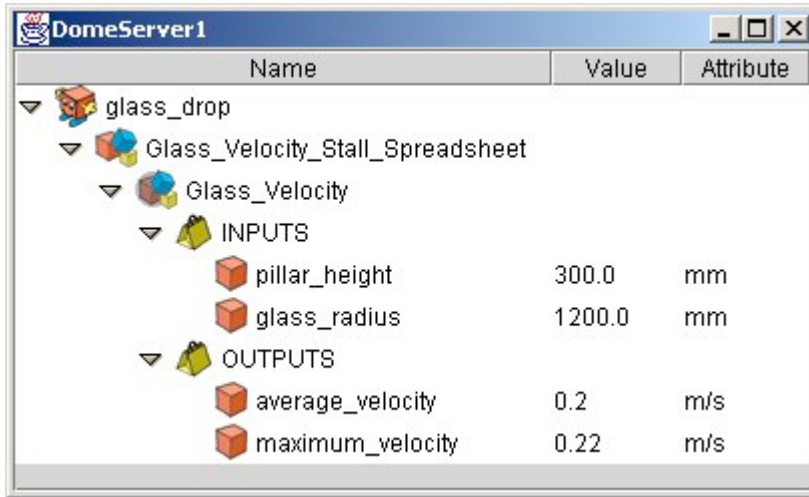


**Figure 6** Creating a parametric interface to the DOME wrapper model.

Next, the glass-velocity engineer can deploy the DOME model and its interface onto a DOME server. This step is analogous to deploying a html file onto a web server, with the exception that one can also define access privileges for each interface and, if desired, define collaborative groups that can share information from the same running instance of the simulation.

Figure 7 shows the Glass\_Velocity interface deployed on DomeServer1 within the workspace glass\_drop. Thus, any client with Internet access and appropriate permissions can enter the Glass\_Velocity interface and remotely drive what-if scenarios using the interface parameters. Within the client user interface DOME automatically organizes parameters into input and output contexts based upon the causal structure of the underlying simulation.

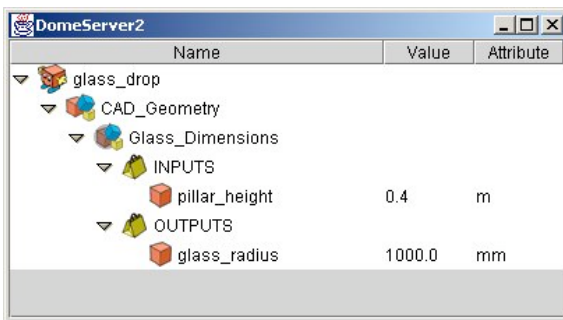
If a client changes an input parameter in the DOME model interface, DOME propagates the change to the spreadsheet and retrieves new values for the output parameters. The back-end connection to the third party application is generated by the DOME wrapper automatically, so the underlying instructions to provide this functionality are completely transparent to simulation builders and deployers.



Name	Value	Attribute
glass_drop		
Glass_Velocity_Stall_Spreadsheet		
Glass_Velocity		
INPUTS		
pillar_height	300.0	mm
glass_radius	1200.0	mm
OUTPUTS		
average_velocity	0.2	m/s
maximum_velocity	0.22	m/s

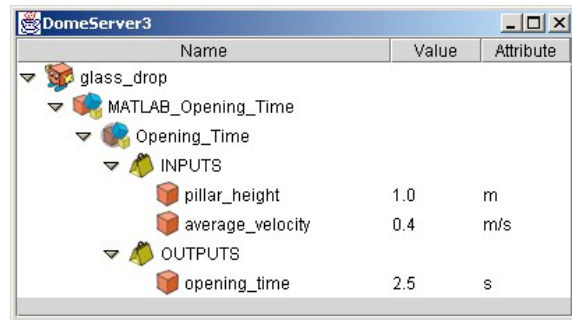
**Figure 7** Deployed Glass\_Velocity interface to the engineer’s spreadsheet.

Similarly, Figures 8a and b show the CAD designer’s simplified Glass\_Dimensions interface (wrapping an I-deas CAD model) and a third engineer’s Opening\_Time interface (wrapping a Matlab simulation). The process for defining and deploying the additional wrapper model interfaces is similar to the spreadsheet example.



Name	Value	Attribute
glass_drop		
CAD_Geometry		
Glass_Dimensions		
INPUTS		
pillar_height	0.4	m
OUTPUTS		
glass_radius	1000.0	mm

**Figure 8a** Glass\_Dimensions interface.



Name	Value	Attribute
glass_drop		
MATLAB_Opening_Time		
Opening_Time		
INPUTS		
pillar_height	1.0	m
average_velocity	0.4	m/s
OUTPUTS		
opening_time	2.5	s

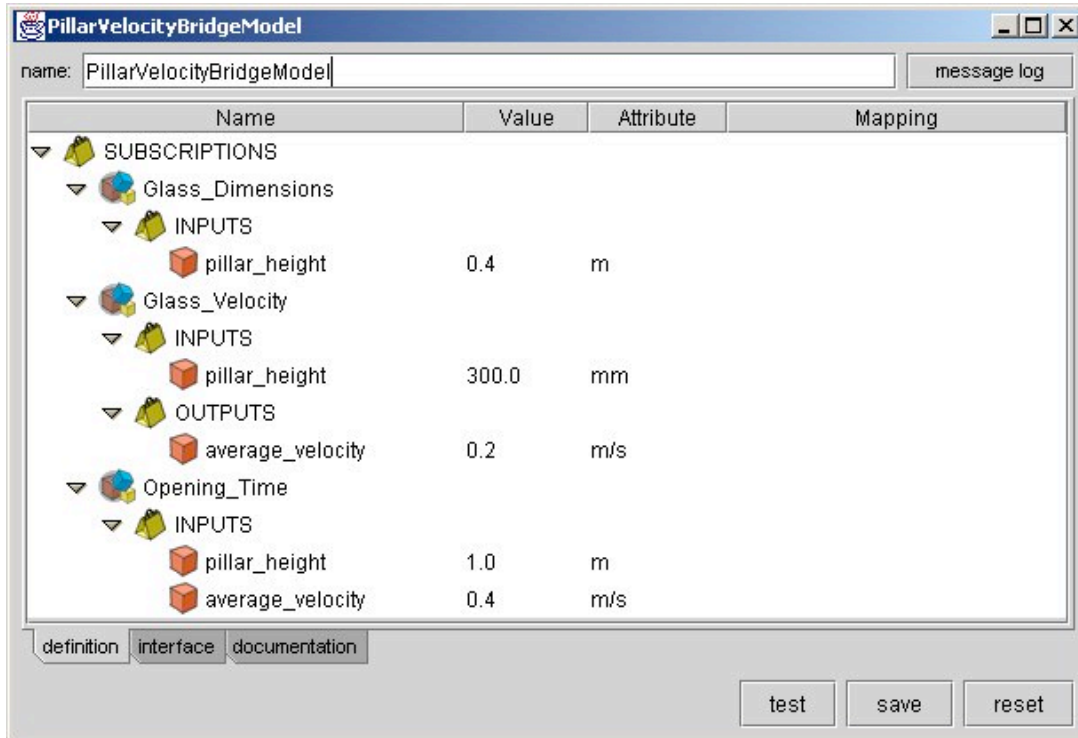
**Figure 8b** Opening\_Time interface.

Interfaces to a variety of heterogeneous simulations are now remotely accessible and operable over the Internet, which is one essential component of the WWSW concept. A second critical element of the WWSW concept is to provide each individual participant with the ability to define system interactions locally, just as individuals can choose to create html pages with links to other pages in the WWW.

In order to illustrate emergent system synthesis in the WWSW, let us assume that the opening-time engineer is interested in studying the effect of changes in pillar height. The

opening-time engineer will do this by creating a new DOME model that subscribes to interface parameters of deployed DOME models.

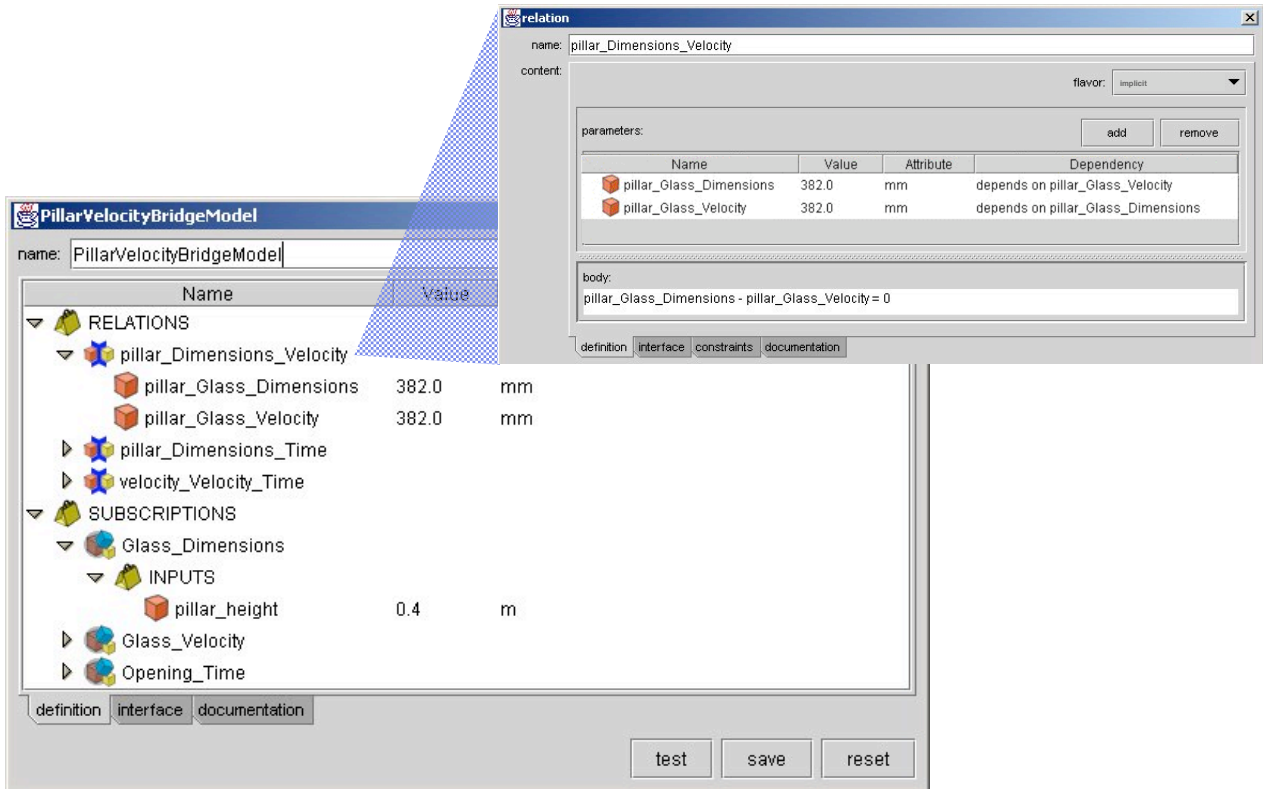
In Figure 9 the opening-time engineer has started to build a DOME model called PillarVelocityBridgeModel. The opening-time engineer has searched the deployed DOME interfaces of his collaborators' simulations, and subscribed to the parameters that are needed for this study—the pillar\_heights from the Glass\_Dimensions, Glass\_Velocity, and Opening\_Time simulation interfaces, and the average\_velocities from the Glass\_Velocity and Opening\_Time interfaces.



**Figure 9** The opening-time engineer building an integrated pillar study simulation using parameter services from the interfaces of other simulations.

The process of making subscriptions involves logging into deployed model interfaces, highlighting the desired interface parameters, and then choosing *subscribe* from a menu option. However, transparent to the user, the subscribed parameters are mapped through a XML-based, bi-directional, distributed communication link.

In Figure 10 the opening-time engineer has added mathematical relationships to the DOME model. DOME is a simulation model-building environment, and thus it is possible to add any number of parameters and relationships (either directly or through subscriptions to other models). It is important to provide this capability within the WSW synthesis environment since interface parameters from different simulations are not likely to be fully consistent, and there may be missing pieces that need to be added within the integration environment.



**Figure 10** Mathematical relationships are added to PillarVelocityBridgeModel.

Three relationship objects have been added and the graphical interface for the one named pillar\_Dimensions\_Velocity has been expanded. The body of the relationship is:

$$\text{pillar\_Glass\_Dimensions} - \text{pillar\_Glass\_Velocity} = 0 \quad \text{equation (1)}$$

The body of the second relationship, pillar\_Dimensions\_Time is:

$$\text{pillar\_Glass\_Dimensions} - \text{pillar\_Opening\_Time} = 0 \quad \text{equation (2)}$$

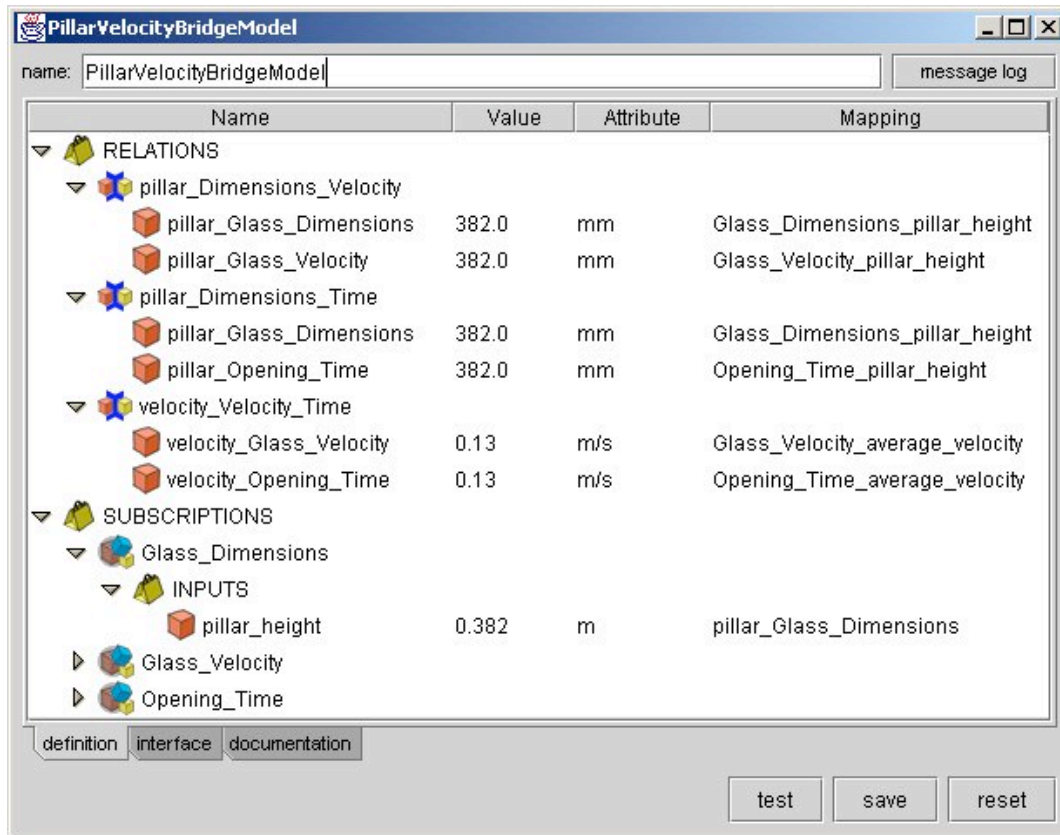
The body of the third relationship velocity\_Velocity\_Time is:

$$\text{velocity\_Opening\_Time} = \text{velocity\_Glass\_Velocity} \quad \text{equation (3)}$$

It should be noted that equations (1) and (2) do not have fixed causality— their input/output structure will change depending upon which parameter initiates a pillar height change. In contrast, the input-output mapping of equation (3) is fixed.

In order to make the relationship definition process as simple as possible, relationship objects check their definitions for dimension consistency and also perform unit conversions as needed.

In Figure 11 the opening-time engineer has completed the model by mapping parameters from the subscribed remote simulations to appropriate parameters in the relationship interfaces.



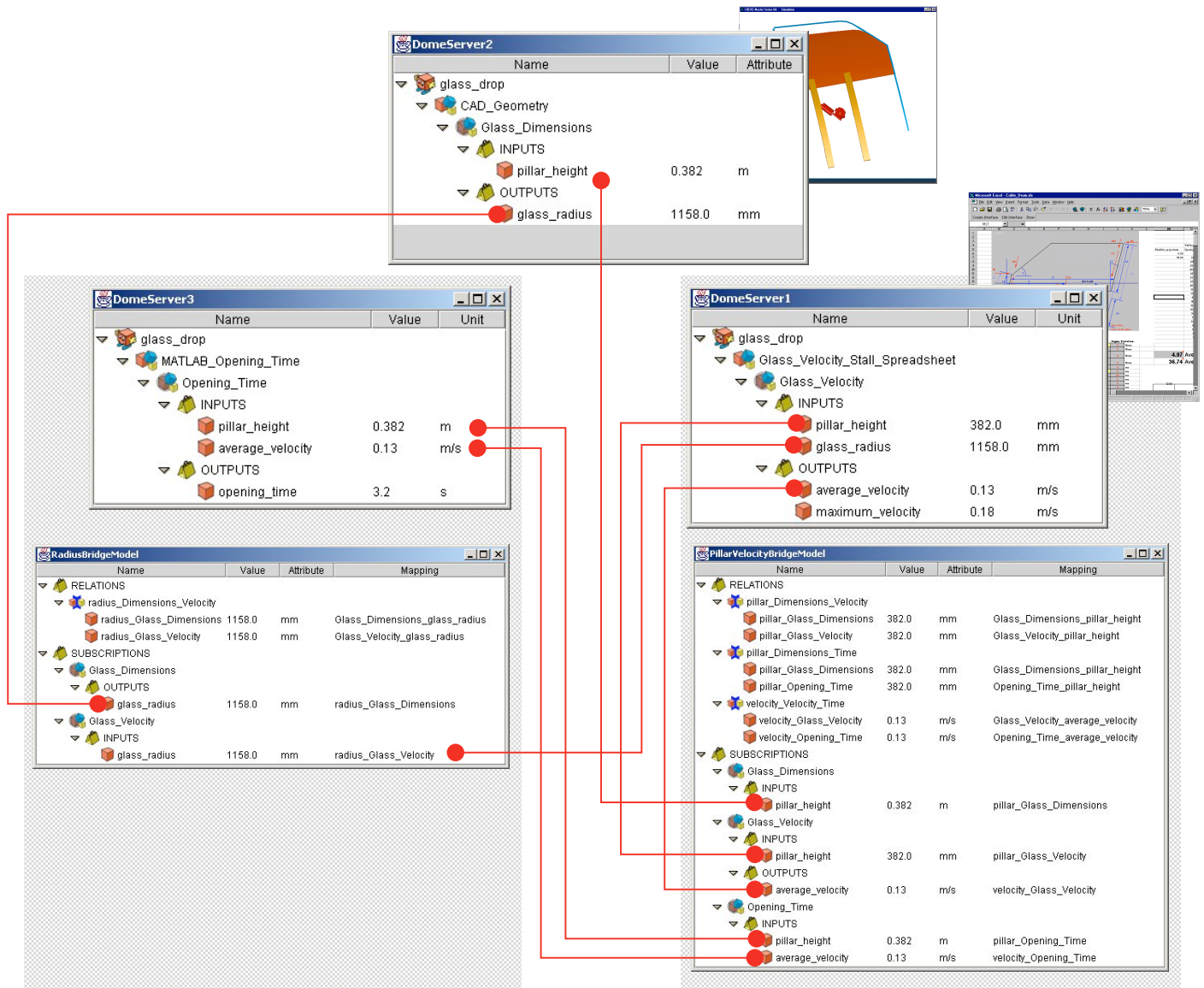
**Figure 11** The completed PillarVelocityBridgeModel with remote parameters mapped to relationship parameters.

Following a similar process, the glass-velocity engineer creates a DOME model subscribing to the radius parameters in the Glass\_Dimensions and Glass\_Velocity interfaces. Within this model a relationship is been defined so that:

$$\text{radius\_Glass\_Velocity} = \text{radius\_Glass\_Dimensions} \quad \text{equation (4)}$$

If the glass-velocity engineer attempts to create additional relationships between pillar\_heights, diagnostic information would be provided indicating that the system is over constrained.

The integration activities of the glass-velocity and opening-time engineers have created the integrated simulation shown in Figure 12. The opening-time and glass-velocity engineers were able to integrate the pieces of the overall simulation that they understood without having to consider the global simulation structure.



**Figure 12** Structure of the integrated system involving five simulations located on three different DOME servers.

Even though the system definition is distributed and there is no centralized control, the simulation network can solve itself in an appropriate manner. For example, if a change in the pillar parameter is made within the Glass\_Dimensions interface, the causal structure of pillar relationships in the PillarVelocityBridgeModel model is fixed on DomeServer1 and enforced until the effect of the pillar change is fully propagated. The pillar relations (equations (1) and (2)) are executed and then the CAD\_Geometry simulation on DomeServer2 runs to obtain a new radius from its underlying I-deas CAD model. Once the new radius is obtained, the RadiusBridgeModel simulation on DomeServer3 computes its relationship (equation (4)) and then the Glass\_Velocity\_Stall\_Spreadsheet simulation executes its underlying spreadsheet model. The velocity\_Velocity\_Time

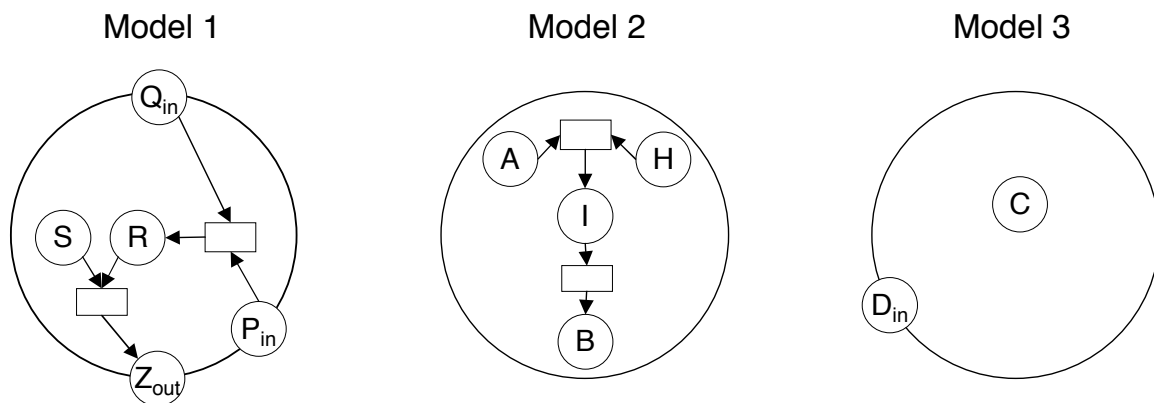
relationship (equation (3)) in model PillarVelocityBridgeModel is then executed, and finally the MATLAB\_Opening\_Time simulation computes the glass opening \_time.

In order to achieve this highly coordinated interaction between the simulations without explicit centralized control, the servers must work together to correctly solve the overall system simulation.

### Federated Parametric Solving in an Emergent Simulation Network

There are many aspects that need to be addressed to provide a useful integrated simulation environment for design synthesis (Abrahamson, 1999). One key component of the WWSW is a forum for publishing and subscribing to parametric simulation services. This functionality can now be implemented using a number of different protocols and technologies (e.g., XML, SOAP, XML-RPC, J2EE, JMS, CORBA, JINI, eSpeak, or .Net, (Fontana, 2001)). However, a second key aspect of the WWSW, allowing individuals to declare local mathematical relationships between parameters while the distributed network takes responsibility for correctly propagating change and maintaining mathematical consistency, is also needed to enable emergent integrated-simulation synthesis.

In this section an approach developed in the DOME project for federated parametric solving of distributed simulations interacting through algebraic relationships is described. Figure 13 shows a scenario involving three different simulations with interfaces on different servers. Boxes depict algebraic relationships, and nodes represent parameters or a self-contained set of models collapsed into a super-node. Interface nodes on the boundary of a simulation are accessible for subscription by other simulations, but the parameters and relationships within each simulation are not exposed to other simulations because of proprietary issues.



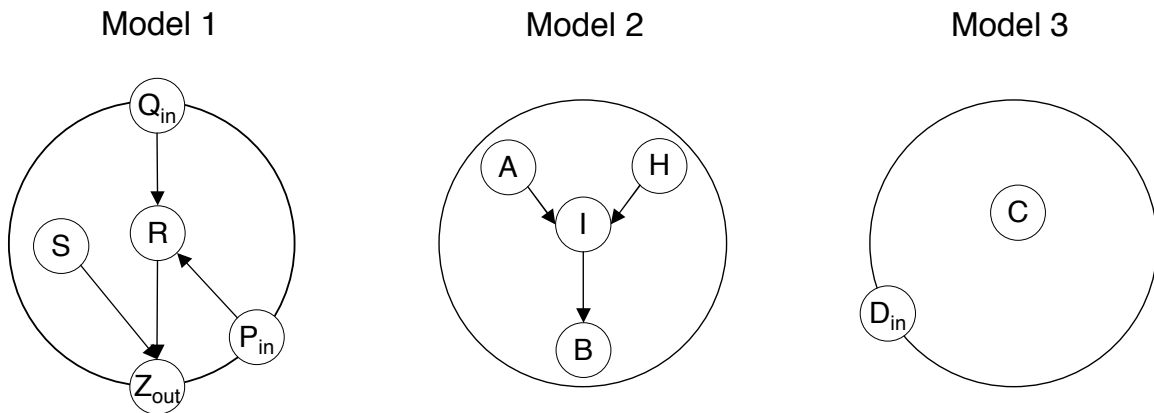
**Figure 13** Scenario involving simulations on three different servers.

Graph-theoretic approaches already exist for evaluating and solving algebraic constraint relationship networks, detecting over-and-under constrained systems, and identifying redundant and conflicting constraints (Serrano and Gossard, 1992). More recently, Park and Cutkosky have developed graph formalisms and solving techniques for dependencies



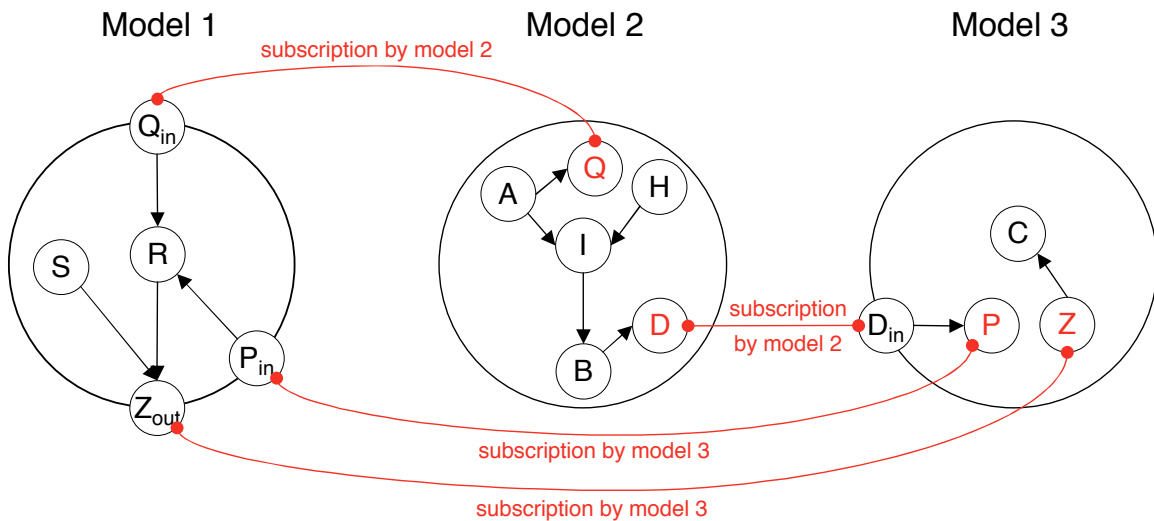
in collaborative simulation networks (Park and Cutkosky, 1999). Additionally, Ueberle has applied a variety of techniques for solving relationship cycles specifically in the context of the DOME project (Ueberle, 2000).

In this section we focus on federated solving coordination between different servers that allows each individual server to solve their local graph efficiently within the context of the larger system without requiring its full details. The discussion assumes that within each server all information needed to solve the local relationship network is available and the network has been reduced to an acyclic directed graph. The directed graphs maintained by local solvers for the example scenario are shown in Figure 14. Edges are used to depict relationships.



**Figure 14** Corresponding graph structures that are solved by each individual server.

Figure 15 illustrates an integration scenario where the three simulations have been connected through interface subscriptions and relationships. The bi-directional subscription mappings are made through independent modeling actions on servers from models 2 and 3.



**Figure 15** Simulations connected through subscriptions.

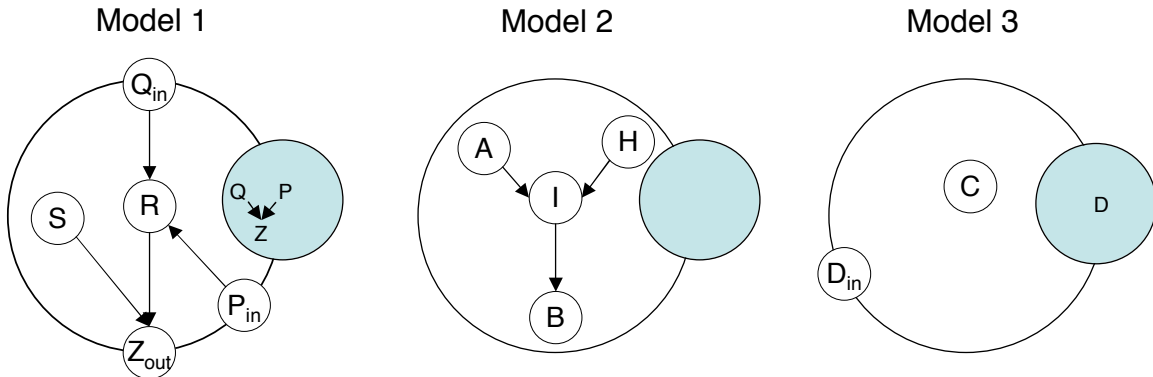
The illustration in Figure 15 assumes that servers do not reveal any information about internal causality during the subscription process—only interface parameters are visible to other simulation servers. Thus, for example, the solver on server 1 maintains the graph representation shown in Figure 14 only and is not aware that Q and P are correlated. Consequently, if parameter A changes in server 2, the simulation on server 1 will execute twice (once when Q changes and once when P changes). Likewise, the relationship mapping Z to C on server 3 will receive two values for Z.

Although this blind messaging approach fully protects the proprietary internal structure of the simulations on each server, it leads to the unnecessary execution of many relationships. This inefficiency can become overwhelming and correlated changes processed sequentially may place simulations into infeasible intermediate states. Additionally, it is not possible for the simulation network to diagnose cycles or constraint inconsistencies.

In order to overcome these problems, most integration tools provide an environment for defining a controlling meta-model or workflow model that sequences the execution of the individual relationships or distributed simulations. These overarching system models are created using explicit, procedural system definition approaches and thus limit the application of integration tools in large, evolving systems that involve heterogeneous simulations and proprietary models.

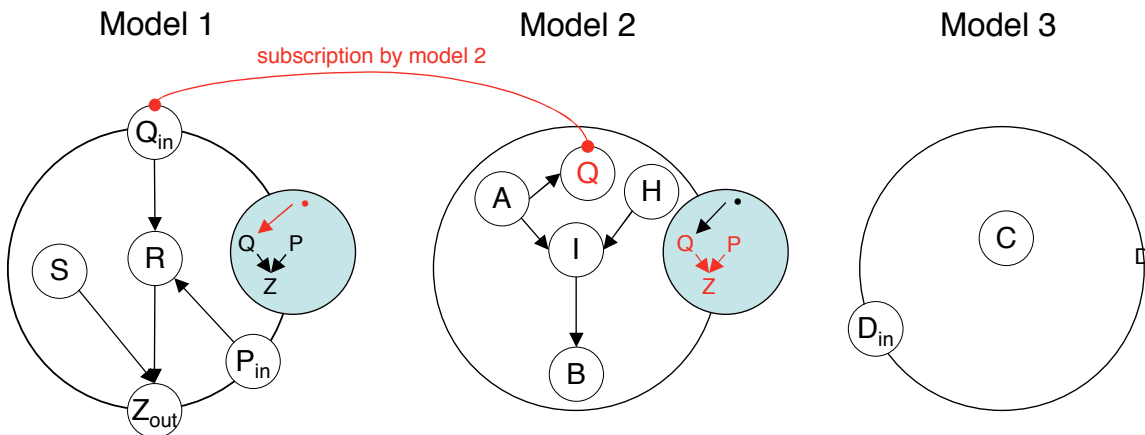
The solvers on each server must share some information if they are to correctly respond internally to external changes, but in order to achieve emergence this information should be obtained in a manner that allows individuals to declare parametric interactions between simulations without writing explicit models and procedures to solve the overall system.

Each simulation server generates and provides a causal map in its interface that relates inputs to outputs. As shown in Figure 16, only the mappings between inputs and outputs are exposed in the interface. The internal structure of the simulation remains concealed. These interface I/O causal maps can be generated automatically for simulations built directly within the DOME environment, and statistical tools have been developed (Kim, 2001) to infer I/O causal mappings for DOME interfaces to black-box simulations.



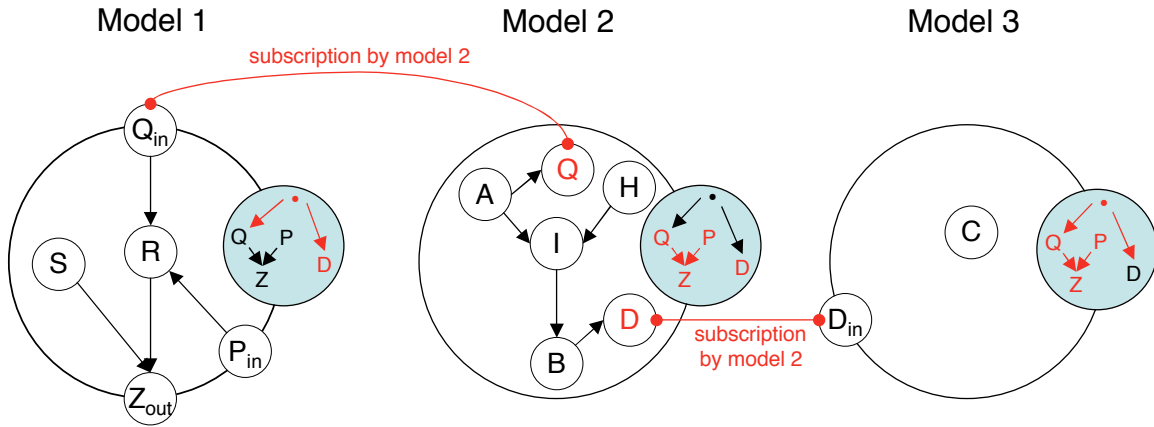
**Figure 16** Causal maps between inputs and outputs published in simulation interfaces.

In Figure 17 the builder of model 2 has subscribed to  $Q$  from the interface of model 1 and then defined a relationship causing  $A$  to drive  $Q$ . During the subscription and relationship definition process the two simulation servers also exchange causal information and update their interface causal mappings. Initially, model server 1 passes its I/O causal mapping to subscribing model 2 and model 2 adds this information to its own interface. Later, when the builder of model 2 relates  $A$  to  $Q$ , this new information is added to the model 2 I/O causal mapping and then propagated back to the subscribed model 1. Local solvers are responsible only for executing the black elements in the I/O causal mapping.



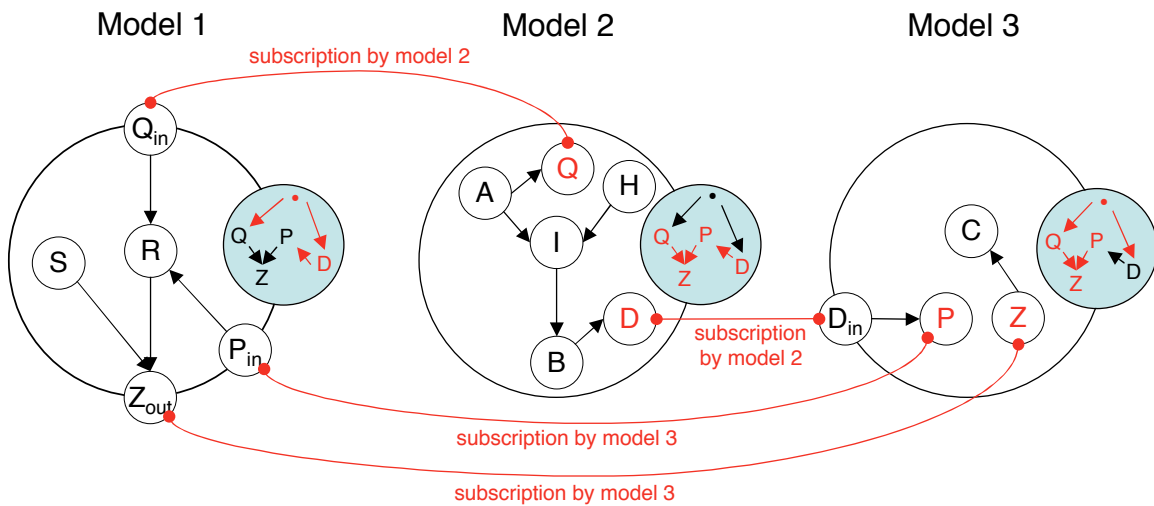
**Figure 17** Interface causal maps are exchanged as part of the subscription process. Model 2 has subscribed to  $Q$  in model 1 and related it to  $A$ .

Next, the builder of model 2 subscribes to  $D$  from model 3 and defines a relationship to drive  $D$  with  $B$  (Figure 18). Model 3 passes the I/O causal mapping for  $D$  to model 2, and model 2 updates its own interface causal map. When  $D$  is related to  $B$ , the model 2 server updates its I/O causal map to incorporate the new relationship with  $D$  and then passes this information to its subscribed models (both model 1 and model 3). If models 1 or 3 were subscribing to parameters in other models, they would likewise propagate the updated information.



**Figure 18** Model 2 subscribes to D in model 3 and relates D to B.

Finally, in Figure 19 the builder of model 3 has subscribed to both P and Z from model 1, and related P to D and Z to C. Following the pattern of the previous examples, model 3 updates its I/O casual map and then passes the new information to its subscribed models 1 and 2. Model 2 will then pass its updated I/O casual map to its own subscribed models 1 and 3. Both models 1 and 3 compare the new information propagated by model 2 with their own maps, and since there is no new information they do not propagate any changes to their subscribed models.



**Figure 19** Model 3 subscribes to model 1, relating P to D, and Z to C.

Like the blind messaging example in Figure 15, each model server is responsible for solving its internal graph structure and executing its internal relationships. However, unlike the example of Figure 15, each simulation server has sufficient information to understand correlations between parameter changes originated by external sources, which enables relationships to be executed in an efficient manner. For example, if Q is changed through the subscription of model 2, the model 1 solver has sufficient information to determine that it should not execute the relationship that calculates R until P also changes.

The process of automatically propagating I/O causal maps allows the independent solvers associated with each model server to coordinate and operate in a federated manner. This allows a system model to emerge as model builders actively subscribe to parameters in simulation interfaces and declare relationships without considering how interactions are solved in response to parametric changes in the integrated system. The federated solving architecture ensures that computational efficiency is not sacrificed and the ability to detect degenerate causal structures is not lost in the emergent simulation synthesis environment.

### **DOME Pilot Applications**

Many of the practical challenges associated with integrated design simulation are not apparent in small, contrived problems so several pilot applications have been conducted with industry sponsors throughout the development of the WWSW concept. Several pilot applications are summarized in Appendix A.

Although the pilot studies have shown clear benefits through the ability to rapidly evaluate complex what-if scenarios, it is difficult to prove the hypothesis that difficulty in building integrated simulations during design synthesis arises because the traditional explicit system definition and integration processes do not match design synthesis activities. However, literature on integrated design simulation attests to the difficulty of applying explicit, procedural, system-definition techniques.

It is also difficult to prove that an emergent simulation integration process with federated solving, as enabled by a WWSW DOME infrastructure, will reduce integration barriers. However, experiences in pilot applications support this belief.

Publishing and integrating 21 heterogeneous distributed simulations for the Ford Moveable Glass System (glass drop) pilot required less than one person-month using an emergent synthesis approach (Abrahamson *et al*, 2000). The integrated simulation required the distributed coordination of some 1500 design parameters. Although there is not a benchmark for building the same system using a traditional definition approach, the authors have observed several similar or smaller size integrated simulation environments that required on the order of person years to construct. A 1999 torpedo design pilot with the US Navy integrating legacy FORTRAN simulations, Matlab models, and CAD geometry models required only 12 hours of integration effort, whereas previous integration efforts of similar scale required several person months (Harrigan, 2000). Similarly, in the 2001 Boeing pilot, preliminary integration of pre-existing Mathematica, Excel and Matlab simulations for predicting the properties of carbon fiber lamina and laminates required only four hours.

### **Conclusions**

The potential benefits of mathematically predicting and analyzing the integrated behavior of product concepts throughout the design synthesis cycle include reducing development time, reducing cost, and improving quality.

Many academic researchers and companies have attempted to develop integrated simulation environments, and it has been consistently observed that significant difficulties arise because of the large scale, complexity, rate-of-change, heterogeneity, and proprietary barriers associated with product design synthesis. Integration efforts have had limited success.

The majority of existing integration environments revolve around technologies, such as distributed communication protocols, integrated data model representations, and ontologies. However, the systems do not question the status quo of defining and coordinating integrated simulations using explicit procedural modeling techniques.

We propose that design synthesis is an emergent system definition process. Since it is very inefficient to represent and structure emergent processes using explicit procedural modeling techniques, a new method for emergent integrated simulation synthesis is suggested. The struggles experienced within many companies to represent product function using a system engineering V may not be attributable to a lack of cooperation on the part of designers and engineers.

Inspired by a vision of the WWW as an emergent information-network building environment, a World-Wide Simulation Web concept is proposed as a model for an emergent, integrated simulation environment. Participants are enabled to make interfaces to their sub-system simulations parametrically operable and accessible over the Internet. Participants can also make connections between parameters in different simulation interfaces and create additional models that bridge interfaces. Each participant defines local relationships between interface parameters in a declarative fashion without having to determine the procedural order in which relationships need to execute for the overall integrated system to operate correctly.

The DOME project has developed a software infrastructure for the purpose of refining and testing emergent integrated simulation concepts. A federating solving mechanism has been developed so that local solvers can respond to external changes in a manner that is consistent with the overall system structure, even though there is no centralized coordination of the overall system. The same mechanism can also be used to automatically inform individuals about the structure of the emerging system. Several pilot studies have been conducted, and results support the belief that a decentralized emergent process for building integrated simulations resolves many of the traditional integration difficulties.

Although the traditional system definition methods work well for clearly resolved consolidated problems, the need for engineering emergent systems is growing as society's expectations of products and technology grows. Currently, it is difficult to predict the integrated behavior of an automobile, but even so, there is an increasing demand that designers foresee infrastructure, urban environment, societal and sustainability implications in relationship to design decisions. Clearly it will not be feasible to address the details of such ambiguous integrated design problems using explicit system definition techniques.

## **Acknowledgements**

The authors would like to acknowledge the contributions of the numerous graduate students that have contributed to the development of the WWSW concept, and in particular Francis Pahng, Shaun Abrahamson, Nick Borland, and Steven Kraines for their fundamental contributions.

## **References**

Abrahamson, Shaun; David R. Wallace, Nicola Senin, and Nick Borland, "Integrated Engineering, Geometric, and Customer Modeling: LCD Projector Design Case Study", *Proceedings of the ASME DT Conferences*, Las Vegas, NV, DETC/DFM-9084, September 1999.

Abrahamson, Shaun; David R. Wallace, Nicola Senin, and Peter Sferro, "Integrated Design in a Service Marketplace", *Computer-aided Design*, 32, pp. 97-107, 2000.

Ahmadi, R.; and R. Wang, "Managing Development Risk in Product Design Processes", *Operations Research*, 47, pp. 235-246, 1999.

Bliznakov, P. I.; and J. J. Shah, "Integration Infrastructure to Support Concurrence and Collaboration in Engineering Design", *1996 ASME Design Engineering Technical Conferences*, Irvine, Ca, 1996.

Bush, Vannevar; "As we may think", *The Atlantic Monthly*, 176, pp. 101-108, 1945.

Case, M. P.; and S. C.-Y. Lu, "Discourse model for collaborative design", *Computer-Aided Design*, 28, pp. 333-345, 1996.

Christian, Andrew D.; Kamala J. Grasso, and Warren P. Seering, "Validation studies of an information-flow model of design", *Proceedings of the 1996 ASME Design Engineering Technical Conferences*, August, 1996.

Cooper, Robin; and Robert S. Kaplan, "The promise—and peril—of integrated cost systems", *Harvard Business Review*, pp. 109-119, July-August, 1998

Cutkosky, M. R.; and R. Engelmores *et al.*, "PACT: An Experiment in Integrating Concurrent Engineering Systems", *IEEE Computer*, pp. 28-37, 1996.

Cutkosky, M. R.; G. R. Olsen, J. M. Tenenbaum, and T. R. Gruber, "Collaborative Engineering Based on Knowledge Sharing Agreements", *Proceedings of the 1994 Database Symposium*, 1994.

Dabke, P.; and A. Cox, "NetBuilder: an environment for integrating tools and people", *Computer-Aided Design*, 30, pp. 465-472, 1998.

Dong, Qi; *Representing Information Flow and Knowledge Management in Product Design Using the Design Structure Matrix*, M.S. Thesis, Massachusetts Institute of Technology, 1999.

Engineous Software; <http://www.engineous.com/engineous.html>, 2001.

Fiper; <http://www.fiperproject.com/fiper/fiperindex.html>, 2001.

Fontana, John; "Web Service: Where Middleware and XML Converge", *Network World*, <http://www.nwfusion.com/buzz2001/webserv/>, September, 2001.

Hameri, A.; and J. Nihtila, "Distributed New Product Development Project Based on Internet and World-Wide Web: A Case Study", *Journal of Product Innovation Management*, 14, pp. 77-87, 1997.

Harrigan, Peter; *Personal Communication*, US Navy Undersea Warfare Group, HarriganPA@Npt.NUWC.Navy.Mil, 2000.

Kim, C., and Y. Kim, "Internet-based Concurrent Engineering: An Interactive 3D System with Markup", *ASME 18th Computers in Engineering Conference*, 1998.

Kim, Jaehyun; *Causality and Sensitivity Analysis in Distributed Simulation*, Doctoral Thesis, Massachusetts Institute of Technology, 2001.

Laws, David; Lawrence Susskind, Jonna Anderson, Ginette Chapman, Emily Rubenstein, and Jaisel Vagadama, *Public Entrepreneurship: Constitutive Roles and Relationships in Sustainable Technology Development*, Draft Manuscript, Environmental Technology and Public Policy Program, 2000.

Molina, A.; and A. H. Al-Ashaab *et al.*, "A review of computer aided simultaneous engineering systems", *Research in Engineering Design*, 7, pp. 38-63, 1995.

NIST, *IDEFO Standard for Function Modeling*, FIPS Publication 183, <http://www.idef.com/idef0.html>, 1993.

Pahng, Francis; *Personal Communication*, Zionex, Inc., francis\_pahng@zionex.com, 2000.

Pahng, Francis; Nicola Senin, and David R. Wallace, "Distributed object-based modeling and evaluation of design problems", *Computer-aided Design*, 30, pp. 411-423, 1998.

Park, Hisup; and Mark R. Cutkosky, "Framework for Modeling Dependencies in Collaborative Engineering Processes", *Research in Engineering Design*, 11, pp 84-102, 1999.

Phoenix Integration; <http://www.phoenix-int.com/>, 2001.



RDCS; <http://www.mscsoftware.com/services/ies/solutions/index.cfm>, 2001.

Senin, Nicola; David R. Wallace, and Nicolas Borland, "Distributed Object-based Modeling in Design Simulation Marketplace", accepted for publication, *ASME Journal of Mechanical Design*, 2000.

Serrano, David; and David Gossard, "Tools and Techniques for Conceptual Design", *Artificial Intelligence in Engineering Design*, 1, pp. 71-116, 1992.

Sferro, Peter; *Personal Communication*, Advanced Manufacturing Technology Development, Ford Motor Company, psferro@ford.com, 2001.

Slate; <http://www.sdrc.com/slate/index.shtml>, 2001.

Toye, G.; and M. R. Cutkosky *et al.*, "SHARE: a methodology and environment for collaborative product development", *International Journal of Intelligent & Cooperative Information Systems*, 3, pp.129-153, 1994.

Truex, D. P.; R. Baskerville, and H. Klein, "Growing Systems in Emergent Organizations", *Communications of the ACM*, 42, pp. 117-123, 1999.

Coleman, H. J.; "What Enables Self-Organizing Behavior in Businesses", *Emergence*, 1, pp. 33-48, 1999.

Ueberle, Mark; *Computational Strategies for Managing Circular Dependencies in Integrated Product Design Models*, Diplomarbeit, University of Stuttgart, Institut für Flugmechanik und Flugregelung, 2000.

Ulrich, Karl T.; Steven D. Eppinger, *Product Design and Development*, 2<sup>nd</sup> Edition, McGraw-Hill, 2000.

Wang, Priscilla; *Emergent Product Development Process Structures*, M.S. Thesis, Massachusetts Institute of Technology, 2000.

Wellman, Michael; "A Computational Market Model for Distributed Configuration Design", *12th National Conference on Artificial Intelligence*, Menlo Park, CA, AAAI Press, 1994.

Wong, A.; and D. Sriram, "SHARED: an information model for cooperative product development", *Research in Engineering Design*, 5, pp. 21-39, 1993.

Wu, Teresa; Jennifer Blackhurst, and Peter O'Grady, "Integrated Enterprise Concurrent Engineering; A Framework and Implementation", *Internet Lab Technical Report TR2001-8*, <http://www.iil.ecn.uiowa.edu/internetlab/TechnicalReports/TR2001-08.PDF>, 2001.

## Appendix A

Summary of major DOME pilot studies conducted for testing and validation during the development of the WWSW concept.

<b>Date</b>	<b>Pilot Focus and/or Application</b>	<b>Participant</b>	<b>Advancement</b>
1997	Electromagnetic Shielding Design	United Technologies and MIT CADlab	<ul style="list-style-type: none"> <li>• Testing of local object model simulation components and solving</li> </ul>
1998	Co-generative Electric Power Plant Design District Heating Plant Design	Swiss Federal Institute of Technology and MIT CADlab	<ul style="list-style-type: none"> <li>• Preliminary test of a distributed object model simulation</li> </ul>
1998	Distributed Service Integration • Battery Powered Drill Design	CADlab (internal)	<ul style="list-style-type: none"> <li>• Testing of simulation service publish and subscribe concepts</li> </ul>
1998	Heterogeneous Third Party Simulation Integration • Environmentally-conscious Design of Beverage Containers	CADlab	<ul style="list-style-type: none"> <li>• Testing of DOME wrapper model concepts, parametric integration of CAD models, LCA simulations, and spreadsheets</li> </ul>
1998	Integrated Engineering and Customer Modeling • LCD Projector Design	Polaroid and CADlab	<ul style="list-style-type: none"> <li>• First test of all major elements of the WWSW concept on an industrial problem</li> <li>• Estimated reduction of integrated analysis of product variants from one month to under a minute</li> </ul>
1998	New Concept Development within DOME • "Watson Project"	Polaroid and CADlab	<ul style="list-style-type: none"> <li>• Test application of concept to a completely new design under development rather than evolutionary design of existing products</li> </ul>
1999	Optimization Incorporating Legacy Code • Torpedo Design	Navy Undersea Warfare and CADlab	<ul style="list-style-type: none"> <li>• First test of web browser-based second-generation implementation</li> <li>• Verification of emergent system definition process</li> </ul>

**Appendix A** continued.

<b>Date</b>	<b>Pilot Focus and/or Application</b>	<b>Participant</b>	<b>Advancement</b>
1999	Integrated simulation across a supply chain, in-field deployment and testing <ul style="list-style-type: none"> <li>• Ford Moveable Glass System (MGS)</li> </ul>	Ford and MIT CADlab	<ul style="list-style-type: none"> <li>• Validation of WWSW emergent simulation definition concept for a complex system involving 5 organizations, 21 simulations, and 1500 parameters between simulations</li> <li>• First time integrated simulations conducted across Ford supply chain and firewall</li> <li>• Supplier interaction time reduced from 2 week to 5 seconds</li> </ul>
1999-	Application to Diverse Ambiguously –scoped Technology Systems <ul style="list-style-type: none"> <li>• Tokyo Greenhouse Gas Half Project</li> </ul>	Alliance for Global Sustainability	<ul style="list-style-type: none"> <li>• Application independent from CADlab</li> </ul>
1999-2000	Product Platform Architecture Air Conditioner Design	LG Electronics	<ul style="list-style-type: none"> <li>• Application independent from CADlab</li> </ul>
2000-	Application to Eco-Systems Seagrass Eco-Engineering for Carbon Sequestration	University of Tokyo	<ul style="list-style-type: none"> <li>• Application independent from CADlab</li> </ul>
2001-	Application to Geometric Assemblies	Ford and MIT CADlab	<ul style="list-style-type: none"> <li>• Parametric assemblies involving distributed components modeled in different CAD systems</li> <li>• Patent pending on assembly modeling technique</li> </ul>