Aspects of S2

Ardeshir Mahdavi, Mustafa Emre Ilal, Paul Mathew, Robert Ries, Georg Suter
Carnegie Mellon University, Pittsburgh, PA 15213-3890, USA

Key words: building performance simulation, distributed system, internet

Abstract: We present in this paper the essential aspects of the S2 system. This is the internet realization of SEMPER, an active, multi-domain, space-based, object oriented design environment for integrated building performance modeling. The key features of the S2 environment are as follows: A user can access the system regardless of the computer hardware, operating system or the location on a network; geographically distributed users can asynchronously generate a building model through the user interface; this building model can then be simultaneously evaluated with multiple simulation applications running on remote simulation servers; persistent storage is provided for project data and evaluation results; designers using the system have access to multiple libraries that contain building information such as material data, construction types, schedules, and weather data.

1. BACKGROUND

1.1 SEMPER

SEMPER is a multi-aspect prototype design environment that incorporates an object-oriented, space-based building model, with *dynamic* links to different building performance evaluation applications (Mahdavi 1996). It is thereby intended to computationally support the evaluation of buildings across *multiple* performance mandates concurrently, with a view toward achieving total building performance and systems integration.

SEMPER incorporates the unique synthesis of four principles:

1. *A methodologically consistent performance modeling approach through the entire building design process*
   The performance simulation modules incorporated within SEMPER rely on a consistent (first-principles based) modeling of the physical processes relevant to the building's performance. SEMPER currently includes modules for energy, air flow, HVAC, thermal comfort, lighting, acoustics, and life-cycle analysis.
2. *Seamless and dynamic communication between the simulation model and the general building representation*
   This provides on-line building performance feed-back to the user while eliminating the need for explicit definition and updating of the underlying simulation model. SEMPER exemplifies this by dynamically linking the building's general representation (essentially a shared object model) with structurally homologous representations of the simulation modules (essentially the domain object models) (Mahdavi and Wong 1998, Mahdavi et al. 1997). Figure 1 illustrates, for example, the homology-based mapping from the shared building representation to the thermal and air flow simulation domains. The shared building representation embodies topological information on the architectural spaces and their relationships. The energy simulation module utilizes a spatial representation consisting of spatial units (cells) with nodes that define finite control volumes for heat-balance calculations (Mathew and Mahdavi 1998). This nodal representation of the building is configurationally homologous to and is directly derived from the space-based representation of the building in the shared object model. This homology-based mapping mechanism allows for rapid feedback and is therefore a powerful design instrument, since evolving building designs can be made subject to comprehensive parametric studies in multiple domains without having to input the building model separately for each domain.
3. *Comprehensive performance modeling*
   In order to analyze the implications of design decisions within and across multiple performance domains, SEMPER incorporates modules to deal with thermal quality, indoor air quality, visual quality, acoustic quality as well as environmental impact (Mahdavi et al. 1996).
4. *Active design support using a "preference-based" performance-to-design mapping technology*
   Active design support enables a designer to interactively and dynamically explore the relationships between design decisions and performance variables (Mahdavi 1993). The active design support in SEMPER would provide two functionalities: a) the user makes a change in a performance variable in order to see the corresponding changes in building design variables; and b) the user makes a change in a building design variable and observes resulting changes in other building design variables when one or more relevant performance variables are constrained.

The proof-of-concept SEMPER prototype 1 was implemented on UNIX with a Tk-TCL user interface, which uses AutoCAD for geometry input and display. This prototype incorporates two simulation modules for energy and thermal comfort analysis.

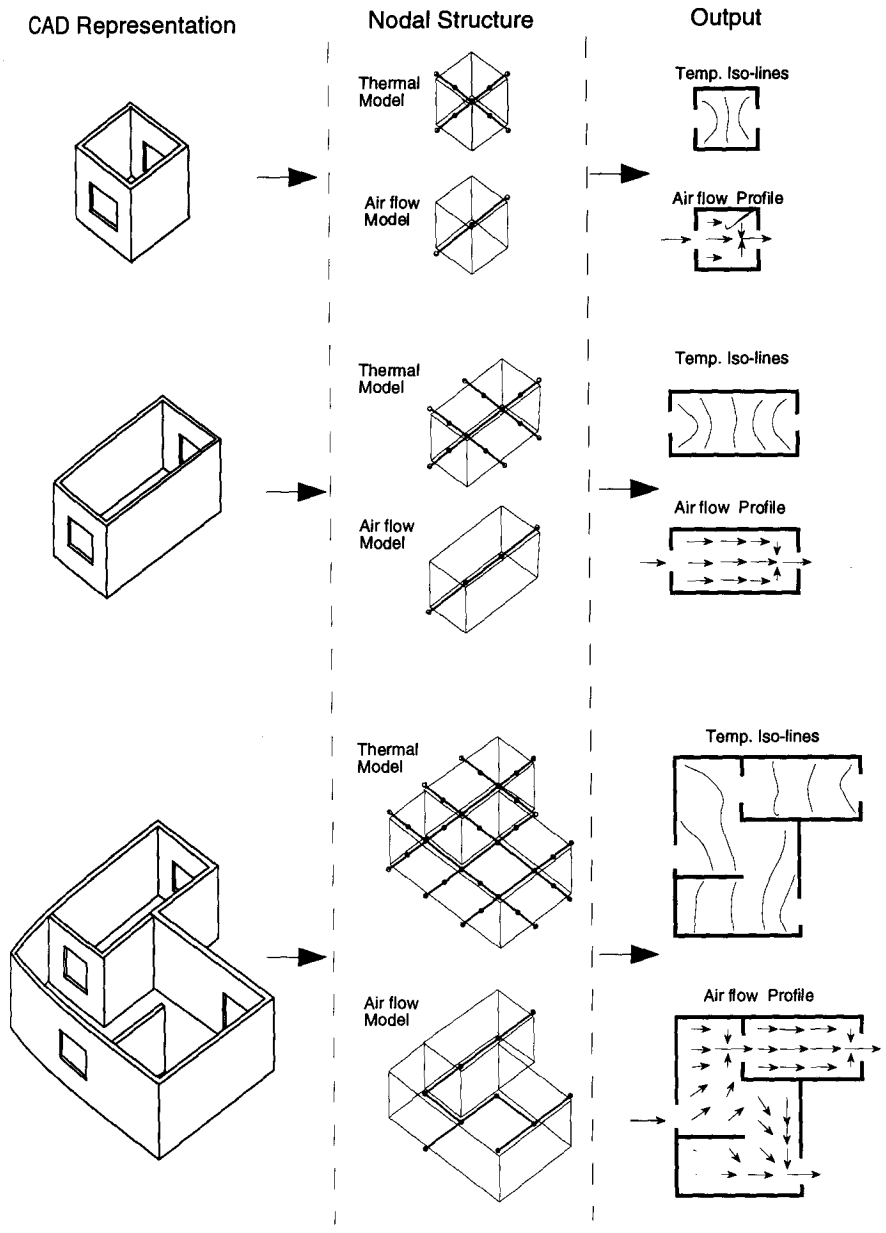

Figure 1. Homology-based mapping of the space-based architectural representation (left) to the nodal building representations in SEMPER's thermal and air flow simulation modules (middle) with their corresponding simulation results (right).

1.2 S2: SEMPER on the Internet

Building on the proof-of-concept prototype 1, SEMPER Prototype 2 (S2) seeks to realize SEMPER on the internet, toward supporting collaborative performance-based building design between geographically distributed users. The various components of S2 (graphical user interface, S2-Kernel, simulation modules, etc.) are distributed over a network, using Common Object Request Broker Architecture (CORBA) as their communication framework. A preliminary implementation of S2 on the internet will involve users at Carnegie Mellon, and two institutions in Singapore - National University of Singapore and Temasek Polytechnic (*Figure 2*).

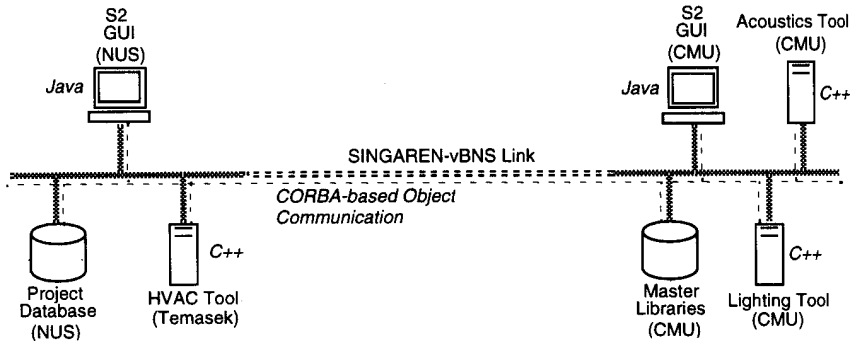

Figure 2. Schematic illustration of components of S2 in the case of two distributed users in Singapore and the U.S. collaborating on a building design/engineering project

2. S2 IMPLEMENTATION

2.1 S2 System Architecture

The S2 system is made up of the following components:

- Graphical User Interface (GUI): Allows users to create and edit building descriptions. It is implemented in Java and includes a simple geometry editor.
- Gateway: The entry point for any GUI. It holds administrative functions and coordinates component interactions.
- S2-Kernel: The application used to instantiate and edit the Shared Object Model (SOM). The SOM reflects the building representational requirements of seven simulation modules. It is implemented in Java.
- Geometric Modeling Engine: A common utility for pure geometric processing.
- Database: Persistent storage for project information as well as shared resources.
- Simulation modules: Applications that can derive their own building representations (DOM's) from the SOM for domain specific processing. The first release of S2 will include modules for thermal analysis, HVAC system simulation, thermal comfort analysis, lighting, and environmental impact analysis.

The component based system architecture of S2 is shown in Figure 3.

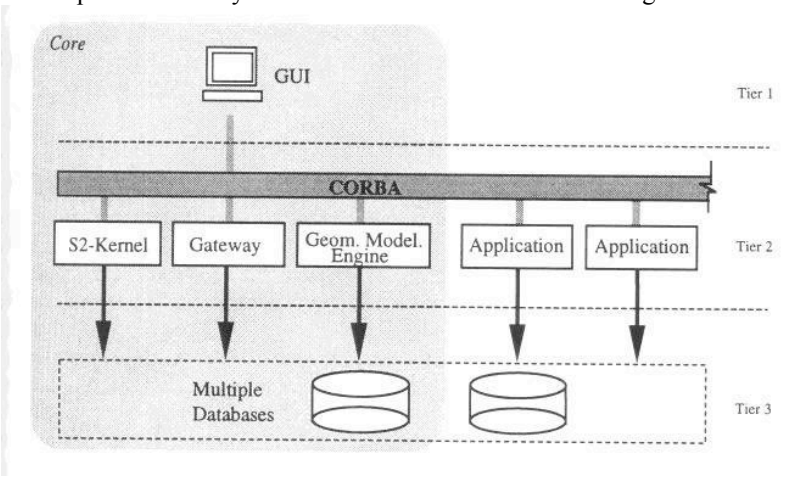

Figure 3. S2 system architecture - components and communication

CORBA was chosen as the communication framework between components. CORBA provides a connection mechanism that is independent of implementation language. This mechanism enables objects implemented in different languages to communicate without the need for any language mapping code. This is an important feature considering S2's implementation which is partially in C++ (Simulation

Modules) and partially in Java (S2-Kernel, GUI). CORBA development also supports a modular structure that simplifies the addition of future components to the system, which is essential for S2.

The GUI is the mediator between the user and S2-Kernel. Whenever a user wants to modify the existing design, the GUI forwards the request to the S2-Kernel, which conducts the requested operation. The result is sent back to the GUI, which displays it.

The Gateway is a connection manager whose URL is published and known by all users. On start-up, a GUI, first contacts the Gateway for authentication. Once the user is authenticated, the Gateway releases an Attendant and assigns it to the GUI. The Attendant acts as a higher level liaison for the GUI. Attendants function as process coordinators and file managers. For example, when the GUI requests the available databases, the Attendant dynamically creates a list of databases which the user has access to and returns the list to the GUI. Similarly, when a user wants to browse the projects inside a selected database, the Attendant conducts the queries and creates the appropriate list. When the user wants to start working with a given project, the GUI again sends this request to its Attendant, which in turn, selects an available server, spawns an S2-Kernel process (which initializes the database connection for the selected project) on it, and connects the GUI with this S2-Kernel.

S2-Kernel is the functional core of the system. It provides access to the SOM which is kept in the database. It also provides all necessary editing mechanisms for it. It populates the SOM and maintains its consistency.

The Geometric Modeling Engine is a library of functions for pure geometric queries. It is a utility available for S2-Kernel and other applications.

When the GUI requests the execution of simulations, its Attendant finds (through CORBA services) the appropriate, available simulation modules and initiates the simulation processes. At the end of such a process, the results are attached to the SOM directly in the database and the GUI-Attendant pair is notified of this event through an event service.

2.2 Shared Object Model

As stated earlier, one of the key features of SEMPER is that a user can input a building description and run simulations in multiple domains without having to manually create domain-specific building representations for each simulation tool. S2 facilitates this through the use of a Shared Object Model (SOM), from which the domain object models (DOM) for each simulation module can be automatically derived without any additional user intervention. The SOM was developed through an iterative bottom-up approach. As such, it is not intended to be a general building model for a broad array of building applications. Rather, it has been specifically developed to reflect the building representational requirements of S2's seven simulation modules. The SOM can, of course, evolve to support additional applications, and additionally, can inform the integration of representations that support simulation into more general models.

The following key features of the SOM bear mentioning:

- The SOM uses a space-based representation i.e. the building is composed of non-intersecting spaces aggregated into sections. This feature was deemed necessary in order to generate the domain representations needed for detailed simulations.
- The primary elements of the building representation each have a geometry (described via an associated geometry class), and a set of non-geometric attributes (described via a 'type' class, referenced by type name). For example, a 'SOMWall' object has an associated "SOMPolygon" geometry object and a SOTConstruction" type object that contains layer and material specifications. The type classes, which essentially represent specification objects, are organized into libraries (e.g. wall construction types, space occupancy types, window shade types, etc.).
- The SOM has a number of "visitor" classes (Gamma et al., 1994) that encapsulate the knowledge for an operation that requires coordination among multiple objects in the SOM. For example, adding a space to the SOM involves a complex series of queries and operations on many objects, requiring a complex traversal of the SOM hierarchy. Visitor classes in the SOM extend from a common superclass so that operation specific visitor subclasses can be added without change to the primary SOM elements.

Table 1 describes the package structure for the SOM. Figure 4 shows the SOM class hierarchy for the primary physical elements.

Table 1. Package structure for the Shared Object Model

Package	**Description**
Semper	Root package - does not contain any classes
Semper.som	Classes that describe the primary elements of the building model - spaces, walls, etc.
Semper.som.visitor	Visitor classes that encapsulate the knowledge for operations that require coordination among multiple objects in the SOM
Semper.geometry	Classes that describe geometric entities used in the SOM (e.g. points, lines, polyhedra, etc.)
Semper.sot	"Type" classes that encapsulate specification information for the primary physical elements of the building model
Semper.util	Utility classes

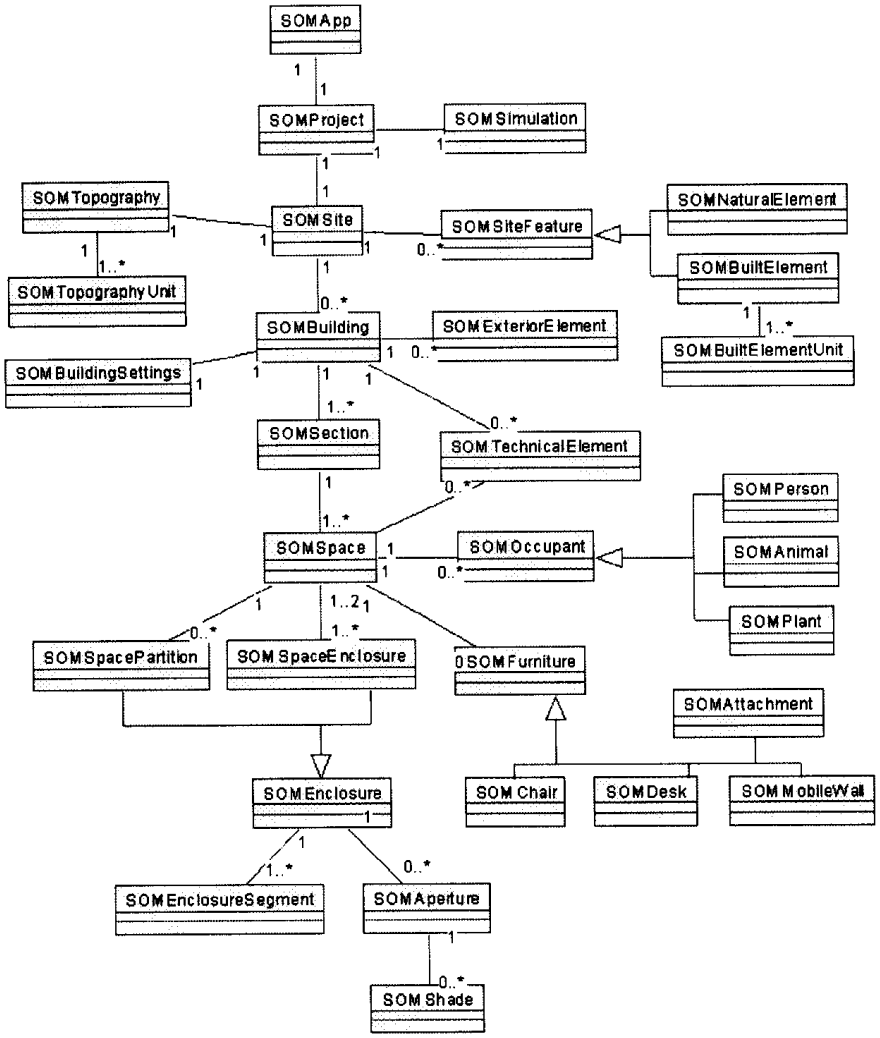

Figure 4. The class hierarchy for primary elements of the SOM

2.3 Graphical User Interface

2.3.1 User interface overview

The system architecture of S2 allows for any number of different interfaces to connect and work with building descriptions that are stored in the database. While any CAD package will be able to connect and update design descriptions, the STS current GUI is aimed at demonstrating core functionality of the overall system. It is modest yet is designed to reflect the main principles of S2.

Building simulation tools typically require detailed input models and often generate enormous amounts of output data. Both the specification of building models and the interpretation of simulation output are tasks that should be supported in an efficient manner by user interfaces. An important aspect of the user interface in S2 is modularization, which is consistent with the overall system architecture. In the 52 environment, designers specify building geometries by drawing spaces in a graphical editor. This editor serves as a starting point for navigating and manipulating non-geometric building attributes such as constructions, materials, and schedules. Similar editors are used in expert design modules to facilitate the communication of more domain-specific information. A simulation setup window allows designers to define space-time boundary conditions for individual as well as groups of simulation modules. After running a simulation, designers browse simulation results in a visualization module, which allows for the navigation of various kinds of simulation data as well as customization of data filters and presentation types.

2.3.2 Input

Designers input building geometries by drawing space-constituting polygons on a canvas. These polygons are communicated to S2-Kernel. In addition to drawing tools, the editor also includes navigation aids for cycling through building sections. Semantic information associated with particular spaces can be accessed by directly selecting a space on the canvas.

2.3.3 Expert design modules

The objective of S2 to facilitate design evaluation at various levels of abstraction is reflected in the interface by the separation of general and more domain specific input functionality. Expert design modules are implemented as plug-ins to the main user interface and are activated only at the user's explicit request. This organization is aimed at reducing the information load on the designer and also allows for increased operational flexibility. Expert design modules may be run as stand-alone as well as part of S2.

2.3.4 Construction and schedule editor

In S2, constructions and schedules are assigned in two ways. Semantic properties may be assigned at the space level, or global building settings for constructions and schedules may be applied to all the spaces in a particular building. This feature is intended to support rapid modification of space aggregates. Enclosure constructions or shading schedules may be modified in one operation at the building level rather than multiple operations at the space level. The only drawback of such an operation is the loss of individual space information. However, the global settings, once they have been applied to a specific space, may be subsequently overridden by designers at the space level without affecting other spaces.

2.3.5 Simulation setup

Simulation setup in an integrated simulation environment such as S2 requires the definition of multiple space-time parameters for each simulation module involved, Individual parameters include grid resolution, time steps, time period, and spatial range. Users would typically select predefined simulation settings for each module and specify only whether modules are run separately or together.

2.4 Domain Object Models

It has been noted earlier that the building representations for specific simulation modules i.e. the domain object model (DOM), is derived from the Shared Object Model (SOM) without any user intervention. This approach has been successfully implemented for domains such as thermal analysis, air flow analysis and environmental impact analysis using homology-based mapping.

However, domains that involve detailed design of physical components, such as HVAC, present a complex set of issues vis-à-vis the relationship between the SOM and the DOM. Indeed, the role of the HVAC analysis within the overall SEMPER environment exemplifies a number of challenging questions in terms of the shared object model, domain object model, and GUI:

- Building energy analysis can be performed at various levels of depth and resolution. From the software engineering point of view, this represents a number of problems. A building model that is too restricted, may allow only for a limited and ultimately less useful set of analysis options. On the other hand, a model that would capture all the requirements of disciplinary analysis may become too large, leading to the classic problems of massive product models. The question is, to what extent should the object class hierarchy of the primary shared building model reflect the requirements of various detailed applications?
- Certain levels of building energy analysis are more relevant to the types of questions that need to be asked and answered by primary building designers (particularly architects). For example, decisions pertaining to the effects of enclosure and glazing, massing, orientation, and natural ventilation, should be covered by the core analysis capabilities of a building design tool. However, analysis of detailed building support systems and controls may be more relevant to the activities of the domain specialist (e.g. HVAC designer). The question is, how will users with different interest and expertise communicate with a multi-aspect software systems that should allow for different level of analysis?

Our recent research findings have lead us to the following position in the case of the HVAC module's relationship to the overall S2 environment:

- A distinction is made between general user (interacting with the S2 SOM) and the specialist (interacting with DOM).
- The type of HVAC analysis that can be initiated at the S2 level by the general user would be limited to default system types. The user selects the spatial domain to be supplied with HVAC, the type of the system, and optionally the thermal zone boundaries. This information is (apart from the simulation results) the only HVAC-relevant specification included in SOM.
- The HVAC domain derives the DOM based on SOM. Beyond SOM's objects, DOM will include domain- specific objects such as generators (e.g. chillers and boilers), distribution components (e.g. ducts and fittings), and terminals (e.g. diffusers and radiators). The basic configuration of these elements is established by HVAC domain's expert system, based on the previously mentioned, limited, HVAC-relevant information in the SOM.
- Once simulation is performed, the structured set of results are sent back to SOM as simulation results. This is the only exception to the otherwise one-way (SOM-to-DOM) information flow in S2. As a general rule, objects that are not defined and generated in SOM are not "visible" to it and cannot be manipulated by it.
- A domain specialist (or a general user interested in other-than-default HVAC analysis) can directly interact with DOM objects thus circumventing the HVAC domain expert. This interaction does not occur via S2's general GUI, but rather via an independent user interface (or a plug-in to the general GUI). Objects and system designs defined by the specialist in DOM are not transparent to SOM, nor are they saved as part of the SOM. They must be saved as part of the DOM and maintained by the specialist. However, simulation results due to such designs can be properly tagged and sent back to SOM as simulation results.

In summary:

- It is not appropriate to include the detailed classes of all applications in a single SOM. The resulting size and complexity is likely to hamper modularity, make interacting with legacy applications difficult, reduce the ease of system modification, and dilute the structural transparency of the software architecture.
- A distinction needs to be made as to the level of analysis appropriate for the general system user. SOM must include all necessary classes for such analysis.
- Detailed designs by specialists need their respective class hierarchies and user-interaction venues. They must maintain their own designs, alternatives, and analyses. A representation of domain-specific classes in SOM, even in informationally reduced form, is not considered appropriate, given considerations of integrity management of mappings between class hierarchies with different levels of abstraction/resolution. As a general rule, design elements that cannot be manipulated via SOM's GUI should not be represented in SOM.
- While various DOMs will not be integrated within a single unified SOM, applications are conceivable that would allow for concurrent viewing of all object information (e.g. their geometric properties), regardless if they are SOM or DOM objects. This integrated display utility would allow for detection of system integration problems (i.e. conflicts between elements of various domains). Active manipulation of domain elements, however, can be only realized via each domain's user interaction mechanisms.

3. CONCLUDING REMARK

The S2 system has been designed, pilot-tested, and as is currently under implementation. S2 is designed to support collaborative performance-based building design between geographically distributed users, over the internet. This is done by modularizing and distributing the components of S2, using CORBA as a communication framework. The S2 system will be further tested and refined as part of a collaborative projects between Carnegie Mellon University and two academic institutions in Singapore.

4. ACKNOWLEDGEMENTS

The authors would like to extend their thanks to the S2 development team which, besides the authors, also includes: Rohini Brahme, Seongju Chang, Beran Gurtekin, Youngsoo Kwon, Prechaya Mahattanatawe, Zhengchun Mo, Vineeta Pal, Nyuk-Hien Wong.

5. REFERENCES

Mahdavi, A., 1996, "Computational Support for Performance-based Reasoning in Building Design", *Proceedings of the CIB-ASTM-ISO-RILEM International Symposium "Applications of the Performance Concept in Building*, Tel Aviv, Israel. Vol. 1, pp. 4.23 - 4.32.

Mahdavi, A., 1993, "Open' Simulation Environments: A Preference-Based' Approach", *Proceedings of CAAD Futures '93*, CMU, Pittsburgh, Pennsylvania, USA. pp. 195 - 214.

Mahdavi, A. - Wong, N. H., 1998, "From Building Design Representations to Simulation Domain Representations: An Automated Mapping Solution for Complex Geometries", *Computing in Civil Engineering; Proceedings of the International Computing Congress*, 1998 ASCE Annual Convention, pp. 1-10.

Mahdavi, A. - Mathew, P. - Wong, N. H., 1997, "A Homology-based Mapping Approach to Concurrent Multi- domain Performance Evaluation", *Proceedings of the Second Conference on Computer Aided Architectural Design Research in Asia: CAADRIA '97* (Ed.: Yu-Tung Liu, Jin-Yen Tsou, June-Hao Hou). Hsinchu, Taiwan. pp. 237 - 246.

Mahdavi, A. - Brahme, R. - Kumar, S. - Liu, G. - Mathew, P. - Ries, R. - Wong, N. H., 1996, "On the Structure and Elements of SEMPER", *Design Computation; Collaboration, Reasoning, Pedagogy: Proceedings of the 1996 ACADIA (The Association for Computer Aided Design in Architecture) conference* (Editors: P. McIntosh and F. Ozel). Tucson, Arizona. PP. 71 - 84.

Mathew, P. - Mahdavi, A., 1998, "High-Resolution Thermal Modeling for Computational Building Design Assistance", *Computing in Civil Engineering; Proceedings of the International Computing Congress*, 1998 ASCE Annual Convention, pp. 522-533.

Gamma, E. - Helm, R. - Johnson, R. - Vlissides, J., 1994, *Design Patterns: Elements of Reusable Object-Oriented Software* (Addison Wesley).