Towards a Virtual Laboratory for Building Performance and Control

Ardeshir Mahdavi
Dept. of Building Physics and Human Ecology
Vienna University of Technology
Karlsplatz 13(270.3)
A-1040 Wien
amahdavi@tuwien.ac.at

Andreas Metzger, Gerhard Zimmermann
Dept. of Computer Science
University of Kaiserslautern
P.O. Box 3049
D-67653 Kaiserslautern
{metzger, zimmerma}@informatik.uni-kl.de

Abstract

The design of a modern building presents a demanding task, as a wide range of aspects (the building architecture and installation, the building control system, and its users) has to be considered. This paper explores the potential and feasibility of a virtual laboratory for building performance and control toward supporting the building design and operation processes.

1. Introduction[1]

Decisions on building design, construction, operation, and decommissioning have wide-reaching implications for people, environment, and economy. This suggests that such decisions must be carefully scrutinized in view of their multi-faceted implications.

The design of modern buildings is a challenging task, because a complex pattern of factors impacts a building's performance. For example, the energy consumption of buildings is influenced by formal and configurational properties (building geometry, massing, orientation, etc.), semantic (attributive) variables (e.g. properties of materials such as density, thermal conductivity, specific heat, thermal and solar emittance, absorptance, transmittance, and reflectance), and contextual properties (such as site and micro-climatic conditions).

Also, the building installation (e.g. heating/cooling generation, distribution, and delivery components), the control systems (processing units and control software), and human behavior have a strong impact on the overall energy consumption.

The situation becomes even more complex, if we consider the interrelationships between various domains (e.g. between heating, cooling, and lighting), or if we consider more comprehensive indicators of building performance (e.g. total environmental impact assessment instead of energy use prediction).

In addition, the design process is complicated by the communication and negotiation requirements of stakeholders from different domains (e.g. inhabitants, facility managers, architects, and control system specialists): Dynamic properties of the building's control system (e.g. the rate of cooling an area in response to user occupancy) requires the consideration of such multi-domain interests.

Further, the number of components that needs to be dealt with poses a problem. For example, the automatic control of a 100 room office building, might require as much as 1000 physical components to be considered.

Last but not least, for designing a modern building, its users (inhabitants, facility managers, etc.) can no longer be regarded as passive entities but have to be considered as *active* persons, which—driven by their own decisions—interact with all parts of the building and the control system. It is a well known fact that the more users are allowed to have control of the building, the more the acceptance of the control system and the perceived comfort increase. The challenging question that remains is, how much and which kind of control is optimal.

Given these circumstances, few would disagree that it is not an easy task to carefully and critically review and evaluate design decisions. We suggest that such evaluation can be supported by conducting experiments in a virtual laboratory for building performance and control.

2. The Virtual Laboratory

A virtual laboratory can be regarded as the simulation and extension of a real laboratory by means of computer support. Therefore, a virtual laboratory allows one, in principle, to evaluate real experiments and operations (e.g. construction of a building, implementation of a control system) *before* they are performed, by applying computer simulation, and it allows one to gain an improved evaluation of real experiments and operations *while* they are performed, by augmenting them with the aid of computational techniques.

A prerequisite for efficiently carrying out virtual experiments is the availability of models of the real laboratory for constructing the respective computer simulations.

Modeling has a long tradition in the architectural design process. From physical scale models to advanced virtual reality representations, architects have always used various media to 'externalize' design ideas [Mahdavi, 1997]. Historically, the main concern of architectural modeling has been visual appearance and constructability (e.g. dimensions, structure, spatial relationships, massing, color, materials).

[1]This paper has been published in R. Trappl (Ed.) *Cybernetics and Systems 2002*. Vol. 1. Vienna: Austrian Society for Cybernetic Studies. 2002. pp. 281–286

However, the increasing complexity of building technologies and the growing awareness of the need for environmental conservation and habitability have led to a broader view of architectural modeling. For example, models now are used to explore the performance of buildings in view of energy consumption, thermal comfort, illumination, and acoustic quality [Mahdavi, 1999]. The computer simulations attained from such models are referred to as *building performance simulators*.

For the software development process, only recently has the modeling of software become broadly applied. These models allow the validation of software properties of the control system before the software implementation actually commences. The computer simulations constructed from these models are called *software prototypes*. In an environment of a real-time building performance simulator and a real-time software prototype, *real* users can interact with the virtual building if meaningful user interfaces are provided. However, to be able to conduct series of experiments with controlled changes, user interaction has to be repeatable. For this purpose, *user simulators* are required, which can be an integral part of building performance simulators.

The purpose of models and simulations is not only to communicate the properties of designs with owners, contractors, and constructors, but also (and primarily) to support a kind of internal dialogue between the designers and their evolving design ideas and solutions. Models help support this dialogical process in many ways. They allow certain features of the design to be emphasized or isolated and thus more closely studied.

Moreover, models allow for performing a large number of parametric operations (variations, alterations, comparisons) that would not be feasible on real artifacts.

The overall simulation model for a virtual building needs to consider the building's architecture and installation, the building's users, the building's control system, and the interrelations between these parts (see Figure 1). Where the control system software is simulated by *control system prototypes* (see Section 5), the remaining parts are simulated by *building performance simulators* (see Section 4). Additionally, the interrelations are established by specialized tools (see Section 6). However, as models abstract from reality, and as the building architecture, installation, or the control system software might partially or completely exist in reality, a mix of both virtual and real parts has to be considered. This also implies that the interaction of real or simulated users must be taken into account.

During a virtual experiment, different activities are executed by *designers*, and can be supported by *tools* (see Figure 1). Stimuli and settings are *injected*, and the building's responses are *measured*. Typical stimuli are real or synthetic weather data, live or recorded human behavior, etc. Settings can be concrete parameters for control strategies, or a specific choice of building materials. The building's responses are commonly measured by changes to physical variables, like temperature or illuminance.

The *evaluation* of these responses usually leads to a *modification* of stimuli, settings or models. Where modified stimuli and settings can directly be injected, modifications to models require new computer simulations to be *constructed*. As an example, an evaluation might identify the problem that a certain room does not heat up fast enough. A first solution is changing the settings for the room's temperature control. If this doesn't lead to the expected behavior, the room's installation might need to be modified (e.g. an additional radiator must be installed).

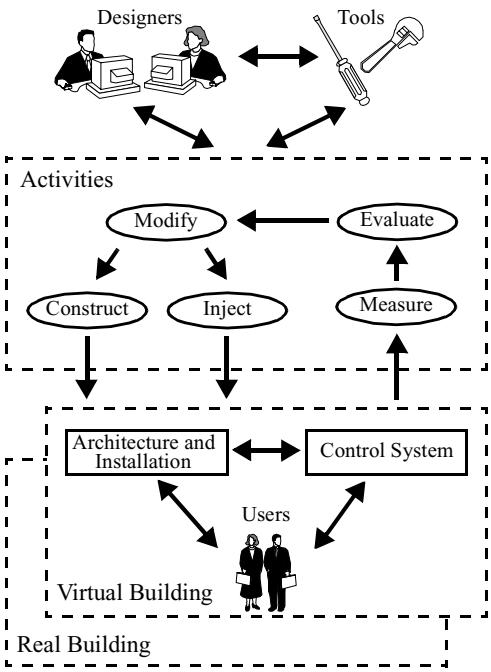

Figure 1: The Virtual Laboratory

For the depicted virtual laboratory to become feasible, models and simulations must be attainable, and the execution of the described activities has to be possible. Also, the efficient construction of computer simulations from models must be possible to provide fast feedback cycles, and thus enable learning effects. The following sections of the paper describe possible approaches for constructing building performance simulators and control system prototypes, and present tool concepts for supporting the activities to be carried out.

3. State of the Art

Virtual laboratories are already successfully applied in many disciplines, like chemistry (e.g. the irYdium Project at Carnegie Mellon University [Yaron *et al.*, 2000]), and physics, where they are commonly employed for educational tasks.

With the advent of electronic computing technologies, many engineers in building-related domains (structure, thermodynamics, acoustics, illumination engineering, etc.) developed the first generation of procedural programs to expedite routine mathematical calculations needed for sizing and evaluating building components and systems.

While initially analytical approaches where the prime candidates for such early applications, numeric methods have become more prevalent recently. Such methods allow, for example, the solution of complex heat and mass

transfer phenomena in the built environment, which previously were outside the reach of traditional manual and analytical methods. Advanced numerically-based methods enjoy currently an ever increasing popularity amongst consultant professionals in building construction and engineering domains. Such advanced methods include, for example, FEM (finite elements method), FDM (finite difference method), FCV (finite control volume method) in statics and heat transfer domains, as well as radiosity-based radiation exchange methods in lighting and acoustics domains.

As the complexity of building control software increases, systematic design methods become a necessity. Current software design methods are often based on well known modeling languages (e.g. the Unified Software Development Process [Jacobson *et al.*, 1999] based on the UML, and the Real-Time Object-Oriented Modeling Method [Selic *et al.*, 1994] based on ROOM, a predecessor of UML-RT). However, these methods often approach a broad range of problem domains, and therefore lack the benefits (e.g. efficiency, exact description of process steps) that can be gained from specialization to a specific domain.

Many existing design methods (e.g. ROPES [Douglass, 2001] based on the UML) adapt the technique of prototyping [Budde *et al.*, 1992] for establishing communication between stakeholders. However, only with the recent advent of operational modeling languages (like SDL [Olsen *et al.*, 1997], or Statemate [Harel *et al.*, 1998]), and respective code generators has the efficient construction of prototypes from software models become possible.

4. Building Performance Simulators

Building performance simulators, when used for a virtual laboratory, have to meet specific requirements. An important requirement is that these simulators respond to injected changes correctly in relation to time and values. Otherwise, the responses of the coupled control system prototype and of the real or the simulated users would not be reliable. Dynamic changes are not only caused by weather data, but also by control system reactions, and user behavior and control. Because of the absence of interfaces for injecting stimuli and settings, and the lack of calculating truly dynamic reactions, most of the existing simulators cannot be directly applied. We have evaluated two approaches to that problem. In the first approach a control system interface for the existing lighting simulation Lumina [Metzger, 1999], which is a subsystem of the SEMPER [Mahdavi, 1999] simulation environment, has been established. In the second experiment a specific simulator for testing control systems was developed from scratch, based on modeling concurrent objects, and code generation.

The main pillars of the conceptual framework for developing building simulators in the SEMPER project are based on first principles-based design support and integration. The first principles-based design support denotes the conviction that, in order to provide effective and reliable feedback to the designer regarding the performance of a design proposal, it is paramount that the actual physical phenomena in buildings are simulated, instead of using approximative and table-based methods.

The integration principle denotes a particular approach to address the input redundancy dilemma often encountered in multi-domain analysis applications. Currently, the professionals interested in prediction and evaluation of a building's performance in multiple domains (energy, environmental impact, economy, illumination, acoustics, structure, etc.) must create idiosyncratic building models for each application separately.

In contrast to this ineffective, time-consuming, and error-prone approach, the SEMPER project demonstrated that a certain level of multi-domain analysis tool application is possible and feasible for a good number of applications during a significant portion of the building delivery process. The SEMPER's approach is based on a representational labor-division between a general building notation scheme (Shared Object Model or SOM), and a number of disciplinary representations (Domain Object Models or DOMs), which in many cases may be inferred from SOM without user intervention [Mahdavi, 1999].

In our concurrent object-oriented approach the basic idea is to map the physical reality as closely as possible into computational models. Since the mapping of physics at the level of atoms into computer models is infeasible, we chose a coarser granularity by modeling building and system objects that show a simple physical behavior. Such objects can be wall elements, windows, radiators, fans, or air volume cells in larger spaces. Objects, like windows, can further be partitioned into a heat transport, a light transmission, and an air flow object.

Because in reality all physical processes execute in parallel, objects are modeled as autonomous processes that execute concurrently, and communicate asynchronously via messages. Data that are exchanged are typically surface values of the objects (e.g. surface temperatures or heat flows of wall elements).

Internally, the objects calculate their dynamic physical behavior using the finite difference method (FDM). Due to the simple physical behavior that these objects expose, the mathematical equations are accordingly very small, and can be solved with ordinary numerical methods.

The described modeling concepts can be regarded as an extension and modification of the modeling method that is presented in Section 5. Therefore, an operational model can be attained, and an automatic generation of the building performance simulator from this model is possible, which already includes the interfaces for data injection.

We have demonstrated that a thermal model for a 27-room floor with simple radiation and air flow can be described in about two person weeks. The attained simulator can be executed in real-time with acceleration factors of up to 100 on an 800 MHz personal computer [Zimmermann, 2001].

5. Control System Prototypes

As mentioned in section 2, the construction of control system prototypes must be carried out efficiently. We suggest that this can be realized by providing an efficient, domain specific method for the modeling of control systems, followed by the automatic generation of prototypes from these models.

The proposed specification method [Metzger *et al.*, 2001] is based on the application of object-orientation to handle complexity, on reuse to gain efficiency and product quality, and on the use of an operational (formal) specification language to enable the automatic generation of prototypes.

As an input, the specification process (c.f. Figure 2) takes the *problem description*, which is divided into the *building description* and a collection of *needs*. The building description contains a description of the building's structure (e.g. floor plans), and the building's installation (e.g. the location and types of sensors). From the building structure an initial *control system structure* can easily be derived by using the aggregation hierarchy of the building's objects as a start (the set of control objects with identical properties is described as a *control object type*).

The needs informally describe the control system from the point of view of the stakeholders. The needs are split into manageable *control tasks* such that they can be assigned to a single control object type. After the control object types have been identified and control tasks have been assigned, strategies for realizing the control tasks are informally given in natural language leading to a collection of *semi-formal control object types*.

From these control object types, *operational control object types* are modeled. The modeling is aided by templates to simplify recurring modeling steps, like specifying communication channels between control object types. We have chosen the Specification and Description Language SDL [Olsen *et al.*, 1997] for that step because behavior can be specified by finite automatons, which are suitable for the control domain, and tools are readily available.

After the formal control object types have been modeled, a control system prototype can be generated. A commercial SDL Tool (Telelogic Tau SDL Suite [Leblanc *et al.*, 2000]) has been successfully applied for this purpose.

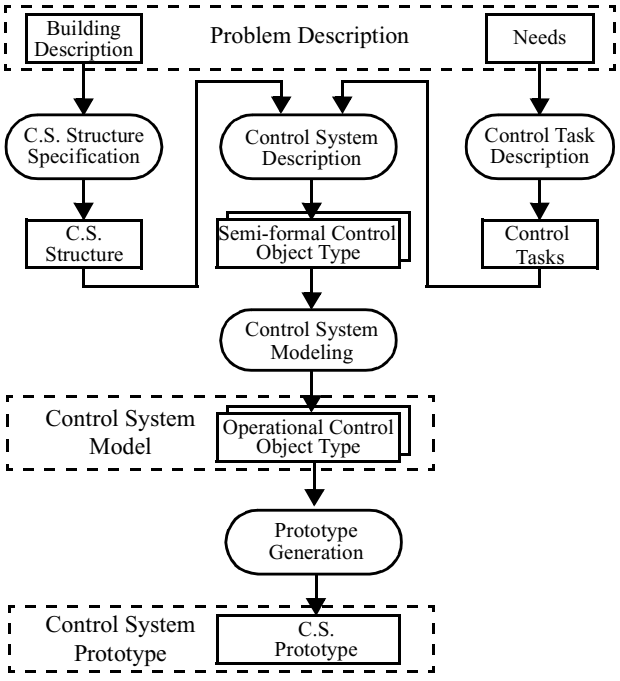

Figure 2: Control System Modeling and Prototype Creation

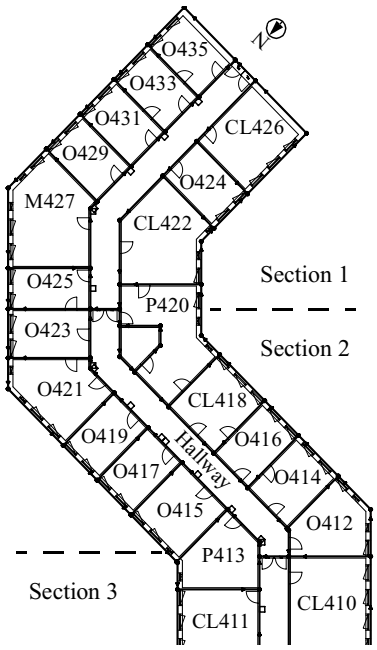

Figure 3: Floor Plan of a University Building

We enable the creation of prototypes as early as possible, by allowing incomplete models as input for prototype generation. For this, missing or incomplete control object types are replaced by *stubs* that roughly mimic the behavior of the respective components [Metzger *et al.*, 2001]. These stubs can automatically be derived from the semi-formal control object types. In addition, this technique can be used for examining selected parts of the control system, where components that are of no interest are replaced by stubs, thus reducing the complexity of the evaluation.

To allow the prototype and the building performance simulator to be interlinked, stimuli and settings to be injected into the control system prototype, and responses to be measured, all prototypes provide an identical interface, through which data to and from control objects can be conveyed. In addition to this interface, the prototypes contain means for tracing internal and external communication between control objects, for recording of internal control system variables, and for registering state changes of the finite automatons (this information is typically written to text files).

The above approaches have been successfully applied in several case studies. We present the most recent one [Queins *et al.*, 1999b], in which the assignment was to develop a lighting and temperature control system for a floor of a university building. This floor, which is separated into three sections, consists of 25 rooms of different types, which are connected by a hallway (c.f. Figure 3). The problem description included a list of 68 different needs. For example, a typical need is: "Shortly before a persons enters a hallway section, the light should be turned on, if necessary."

All needs were split into 126 different control tasks, which were assigned to 37 control object types, which

describe 920 control objects.

The total effort to reach at the formal specification was approximately ten person weeks. From this specification, a control system prototype of about ten megabytes was generated in less than ten minutes on a 440 MHz HP-PA RISC workstation.

During 45 laboratory sessions, which each lasted around 80 minutes, a total of 33 problems in the control system were identified. These problems were corrected by modifying the specification, leading to new control system prototypes. For these laboratory sessions the prototypes were tested with different building performance simulators and real parts of the physical building, which exist as a testbed [Metzger, 2001].

6. Tools

To finally reach at the virtual building, control system prototypes and building performance simulators need to be interlinked. To accomplish this, control objects and objects of the building performance simulator that should exchange data need to be matched (see Figure 4). For this, the control object must provide data that is used by the respective object of the building performance simulator, and vice versa. Typically, these objects represent physical sensors (e.g. *temperature sensor*, and motion detector), and actuators (e.g. *valve*, and luminaire). However, matching corresponding objects at a higher level also seems suitable (e.g. energy consumption of a *room*). Matching such objects becomes complicated as the structure of both systems might differ and even change during development.

Further, the interlinking can be hindered by incompatible interfaces and different notions of time (e.g. where the control system may perform its actions in real-time, the building simulator might need several minutes to calculate results). To solve these problems, a flexible tool, called *interlink*, was implemented [Metzger, 1999]. In addition, the use of such a tool allows an easy exchange of one building performance simulator for another, and permits the coupling of the control system prototype to the real physical building.

After the construction of the virtual intelligent building has been proven feasible, available tools for supporting the activities that are performed during experiments are outlined, and possible concepts for new tools are shown.

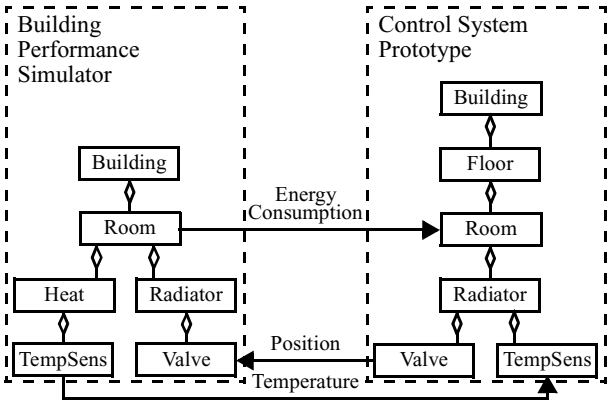

Figure 4: Matched Objects and Exchanged Data

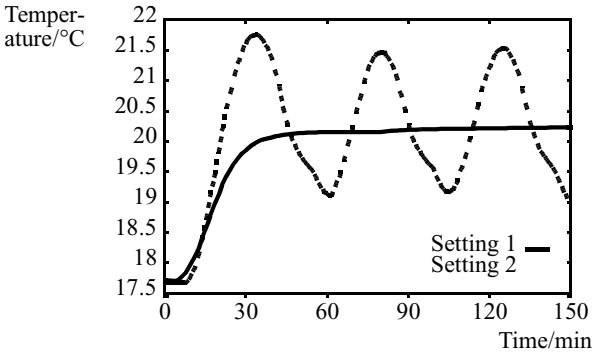

Figure 5: Behavior of Room Temperature Control

As mentioned above, the virtual building allows the recording of dynamic data, like communication between objects. These data tend to become rather huge in number. To illustrate, for the case study introduced in section 5, the files that were recorded during a duration of 50 minutes (real-time) contain one million lines of text, of which 40 000 lines were needed for recording the communication between the control system prototype and the building performance simulator. Therefore, the data need to be condensed to allow a reasonable evaluation.

The data can simply be condensed by extracting information for single objects (e.g. a sensor) and visualizing this reduced set of data. Data extraction can be performed with standard UNIX Shell commands, like *grep* and *awk*, and visualization can be carried out with, e.g. *gnuplot*. Figure 5 shows an example for a (closed-loop) temperature control of a single room for two different settings of control parameters. This simple approach has been successfully applied in several cases, especially for evaluating building performance simulators (see Section 4).

More sophisticated, data can be condensed with specialized tools. One such tool was developed for the analysis of the dynamic behavior (*dynamic analysis*) of the control system prototype [Queins *et al.*, 1999a]. This tool takes the trace of a run of the control system as an input, and besides other information, produces detailed reports on how much communication (counted by the number of messages exchanged) takes place, and by how many control objects a message is propagated. Because of its modular structure, an easy extension of the tool is possible.

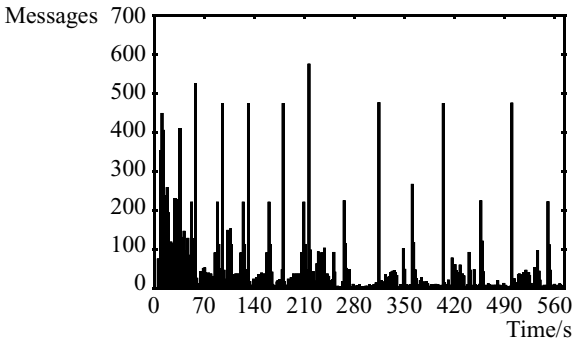

Figure 6: Internal Behavior of Control System

Figure 6 shows an example for the number of internal messages over time.

To produce feasible data for evaluation in the first place, test data need to be injected into the virtual building. This data can be entered automatically or manually. Automatic injection is most suitable for weather data, system settings, and pre-recorded or simulated user behavior. Several tools that read data from file have been implemented and have been successfully applied. A candidate for manual injection is the change of settings that can be performed by the designers at any time during a 'run' of the virtual building. To support this, several variations of tools have been developed.

7. Conclusion and Perspectives

This paper explored the feasibility of a virtual laboratory for building performance and control by presenting approaches for the construction of building performance simulators and control system prototypes.

Moreover, it was shown that basic experimental activities (such as data analysis and injection) can be supported, providing a basis for more elaborate tools in the future. For example, instead of visualizing results as simple graphs, computer simulators for spatial visualization could display physical values (like temperature) directly on rendered images.

Further, the evaluation of data can be automatically performed by tools (e.g. the check if the virtual building conforms to test cases can be automated). Using the results of this automatic evaluation, an automatic optimization of building properties (like building geometry, or parameters for control strategies) may become possible.

References[1]

[Budde *et al.*, 1992] R. Budde, K. Kautz, K. Kuhlenkamp, and H. Züllighoven. *Prototyping: An Approach to Evolutionary System Development*. Springer-Verlag, Berlin, Heidelberg, 1992.

[Douglass, 2001] B.P. Douglass. *Doing Hard Time: Developing Real-time Systems with UML, Objects, Frameworks and Patterns*. 4th Print. Addison-Wesley, Boston, 2001.

[Harel *et al.*, 1998] D. Harel, and M. Politi. *Modeling Reactive Systems with Statecharts: The Statemate Approach*. McGraw-Hill, New York,1998.

[Jacobson *et al.*, 1999] I. Jacobson, G. Booch, and J. Rumbaugh. *The Unified Software Development Process*. Addison-Wesley, Reading, Mass,1999.

[Leblanc *et al.*, 2000] P. Leblanc, A. Ek, and T. Hjelm. Telelogic SDL and MSC tool families. *Telektronikk*, 96(4):156–163, 2000.

[Olsen *et al.*, 1997] A. Olsen, O. Færgemand, B. Møller-Pedersen, et al. *System Engineering Using SDL-92*. 4th Edition, Amsterdam, North Holland, 1997.

[Mahdavi, 1997] A. Mahdavi. A Negentropic View of Computational Modeling. In *Proceedings of the Second Conference on Computer-aided Architectural Design Research in Asia: CAADRIA,* pages 107–122, Hsinchu, Taiwan, 1997.

[Mahdavi, 1999] A. Mahdavi. A Comprehensive Computational Environment for Performance-based Reasoning in Building Design and Evaluation. *Automation in Construction*. 8:427–435, 1999.

[Metzger, 1999] A. Metzger. An Interlink of Building Control System Prototypes and the Lighting Simulation Lumina. Report 8/99, Dept. of Computer Science, University of Kaiserslautern, Germany, 1999.

[Metzger, 2001] A. Metzger. Ein flexibles Testfeld für Experimente im Bereich der Gebäudeautomation und -simulation. Report 4/01, SFB 501, University of Kaiserslautern, Germany, 2001.

[Metzger *et al.*, 2001] A. Metzger, and S. Queins. A Reuse- and Prototyping-based Approach for the Specification of Building Automation Systems. In A. Schürr, editor. *OMER-2 Workshop Proceedings*, pages 3–9. Herrsching, Munich, 2001.

[Queins *et al.*, 1999a] S. Queins, B. Schürmann, and T. Tetteroo. Bewertung des dynamischen Verhaltens von SDL-Modellen. Report 9/99, SFB 501, University of Kaiserslautern, Germany, 1999.

[Queins *et al.*, 1999b] S. Queins, and G. Zimmermann. *A First Iteration of a Reuse-Driven, Domain-Specific System Requirements Analysis Process*. Report 13/99, SFB 501, University of Kaiserslautern, 1999.

[Selic *et al.*, 1994] B. Selic, G. Gullekson, and P.T. Ward. *Real-Time Object-Oriented Modeling*. John Wiley & Sons, New York, 1994.

[Yaron *et al.*, 2000] D. Yaron, R. Freeland, D. Lange et al. Using Simulations to Transform the Nature of Chemistry Homework. *Summer 2000 CONFCHEM*, Online Conference, 2000.

[Zimmermann, 2001] G. Zimmermann A New Approach To Building Simulation Based on Communicating Objects. In R. Lamberts, et al., editors. *Seventh International IBPSA Conference Proceedings*. Vol. 2. pages 707–714, Rio de Janeiro, Brazil, 2001. IBPSA.

[1] Links to documents that are available online can be found at: http://wwwagz.informatik.uni-kl.de/staff/metzger/EMCSR