Design Process-Based Interoperable Software

MARK J. CLAYTON
Texas A&M University

INTRODUCTION

Researchers and software developers are currently developing the next generation of CAD software for the Architecture/Engineering/Construction (AEC) industry. In several initiatives to develop new software, the key strategy has been to enumerate, classify and describe the components of a building. Researchers then attempt to build a consensus upon the definitions in the expectation of sharing information about a building by using the agreed-upon definitions. However, rather than a consensus upon components, there may be advantages to building a consensus upon an understanding of the design process. Comprehensive building design support software could be based upon a model of the designer's cognitive processes instead of a simulation of the material world.

A prototype software system, the Semantic Modeling Extension (SME), has demonstrated that a cognitive model of design is a viable alternative foundation for the next generation of CAD software. SME has proved to be usable by designers in solving a controlled design problem with a restricted set of tasks.

THE NEXT GENERATION OF CAD SOFTWARE

Researchers and developers envision a next generation of CAD software for the AEC industry that is comprehensive in scope yet allows for exchange of design information among the software products of many vendors.

Comprehensive

While the use of computers has become widespread in the AEC industry, many of the expected benefits of computerization have been unrealized. Much of the blame for reduced benefits has been attributed to a lack of integration of information and software (Hansen, Johnson and Tatum 1990). The large number of current software products provide poor abilities to share building descriptions among participants, across stages, and between projects. Inefficiencies that result from the difficulty of sharing information manifest themselves as increased time, increased cost, and increased error rates.

It has been widely suggested that computer systems could overcome this obstacle through greater software and data integration. Rather than being special-purpose, stand-alone systems, future CAD systems could simultaneously or automatically collect information and perform computations that are useful to many participants. In this way, a future CAD system could be comprehensive in addressing the needs of the industry.

Interoperable

A second major obstacle that current research and development is addressing relates to the process of developing and delivering CAD software. The AEC industry may be characterized as fragmented into many small business entities, each of which has individualistic practices. Although the fragmentation probably leads to inefficiencies and disadvantages, it may also lead to exceptional responsiveness and creativity. It may not be possible or desirable to impose a monolithic software solution upon the industry even if it is comprehensive in addressing industry information needs. Such a solution may be too expensive for the average small firm, require too many years for development and require conformance to a rigid design method.

One alternative to a monolithic software solution is "interoperable" software. Interoperable software consists of modules that are provided by diverse software vendors but that nevertheless are capable of sharing and preserving information. A blanket organization for the entire industry would establish and enforce the enabling standards. The concept has been clearly articulated by another author (Eastman 1992).

These two objectives for future CAD systems are complementary. Together, they are a vision of a comprehensive, interoperable CAD system. The system can address the information needs of many participants at many stages in the design, construction and operation process. By allowing the end-users to mix and match modules, the system responds to the personal preferences of the individuals who use it and the particular needs of a project for which it is used. The incorporation of standards for information representation will assure that the building description may evolve during a project and may be exchanged among participants who use other software. Re-input of information, loss of information and redundancy of information will be reduced or eliminated by the common representation. All participants, including building owners, designers, constructors and occupants, will profit.

TWO FOUNDATIONS

Creation of comprehensive, interoperable CAD software will require many years of effort. A key task in that effort is to establish what are the fundamental information units and operations upon which the system may be based. These fundamental concepts serve as the "atoms" from which individual software modules may be constructed. Using the terminology of object-oriented programming, the task may be thought of as answering the basic question "What are the objects?"

Answering the question entails considerable risk. A widely adopted new CAD system could lead to changes in the industry on a seismic scale. If the system guides the design process into a particular direction, it could lead to an improvement in overall design quality or a decline. A new, comprehensive system could lead to a shift in authority and responsibility for a building project to those who use the system, dramatically changing the roles of professionals. Of particular concern is that the adoption of standard building representations introduces a risk that the designer's opportunities for creative expression and invention will be reduced. This concern seems most relevant at the early, conceptual stages of a design project. The choices to be made in the near term may have important consequences to the future practice of architecture.

In the spirit of facilitating a wise answer to the question, my research has postulated an alternative foundation for comprehensive, interoperable CAD systems. While much current research has adopted an approach that emphasizes the components of the building, I have explored an approach that derives from a cognitive model of the design process.

Component approach

Much research into how CAD can be made more comprehensive has employed what has been called the "component approach" (Harfmann and Chen 1990). The basic concept is that a model of the physical building is the common, shared denominator of all of the actions undertaken by participants in the design, construction and operation of a building. Research using this approach has emphasized cataloguing the physical and spatial parts of a building. Software that uses the component approach will be interoperable in that it can exchange standard part definitions. Each part definition will include all necessary descriptive data and behavior. The part models may be assembled into a building model that will express the behavior of the building.

The STEP initiative is an international effort to develop a standard for the exchange of product data. Within the STEP initiative is a focus group for the AEC industry. The information classifications suggested by STEP have focused upon geometry description and physical and spatial components (ISO 1993).

The Industry Alliance for Interoperability (IAI) is a consortium of companies in the AEC industry that is developing a comprehensive, interoperable CAD system architecture (Industry Alliance for Interoperability 1996). The IAI is working to create Industry Foundation Classes (IFC) and build a consensus for them. The IFC will focus upon standardization of data formats and access to the data. Initially, they will not address standardization of behavior or performance of building components but only description of component form. The class hierarchies envisioned are mostly classifications of components.

Cognitive approach

In contrast, my research suggests that an underlying model of the design process is an appropriate foundation for comprehensive, interoperable CAD software. In pursuing this theme, I have employed a basic argument from artificial intelligence that software implementations may provide evidence in support of models of human cognition (Gardner 1985, 140). The model of the design process that I have used is illustrated in Figure 1.

The starting point in this model is the analysis-synthesis-evaluation cycle that is often cited by design methods reports (Asimow 1962). *Analysis*, in Asimow's model, is the determination and documentation of needs for the project. *Synthesis* is the invention of a potential solution to those needs. *Evaluation* is the determination of whether a satisfactory solution has been achieved. My research has focused upon the evaluation step in the design process.

Design evaluation appears to involve three kinds of conceptual objects:

- *forms*, that express the geometry and materials of the design solution;
- *functions*, that express the design requirements; and
- *behaviors* that express the performance of the design in the anticipated use of the design artifact.

These definitions have been elaborated in a previous paper and are similar to those used by other researchers (Clayton, Fischer and Kunz 1995; Gero 1990).

Evaluation involves *predicting* behavior values based upon the design form and then *assessing* the satisfaction of functions with respect to the behaviors. If the assessment is that the functions have been satisfied, then the designer may proceed to either a greater level of detail or to documentation and completion of the project. If the assessment is that the functions are unsatisfied, then the designer must return to the analysis step to check the functions and the synthesis step to generate a new form.

Designers by training and tradition typically cluster par-

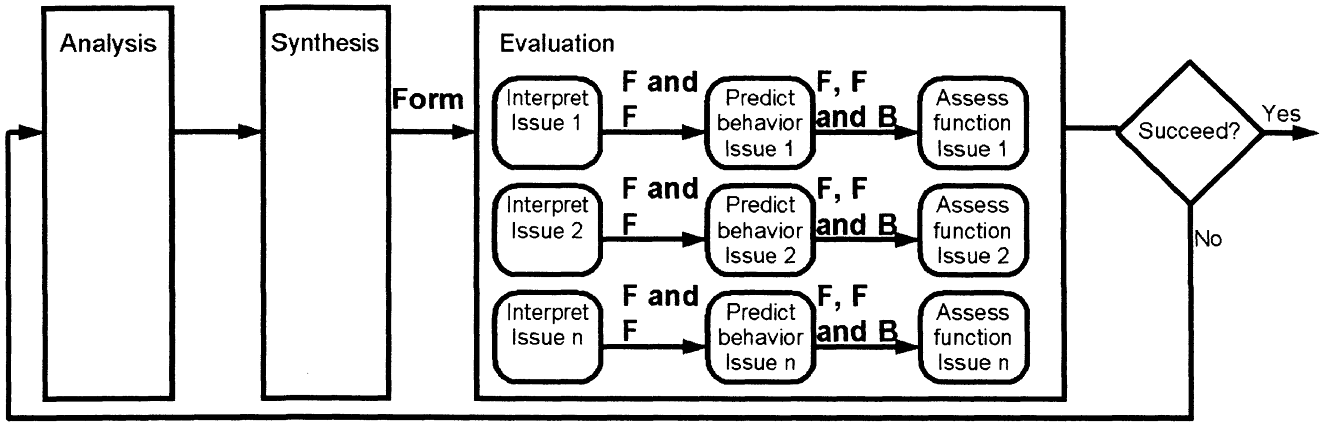

Figure 1, Model of design evaluation process. The synthesis step produces a description of form. During evaluation, the form must be interpreted to associate function with the form for each issue. For each function, behaviors must be predicted using the form. Finally, the functions are assessed by examining the behaviors. The result is a determination of whether the potential solution is successful.

ticular functions into what I call an issue. For example, structural engineering clusters various functions for load resistance and limitation of deflection into a gravity load issue that is often examined independently of other design issues. To accommodate the decomposition of the design synthesis into the issue-based models necessary for evaluation, a designer must perform an *interpret* activity before predicting and assessing. I refer to an interpreted entity in the design as a *feature*.

Of course, this model of the design process is incomplete and lacks detail. However, it has proven adequate for guiding the implementation of software. Thus far in the research, this model of the evaluation process appears to be general. The reification of the concepts of form, function and behavior in the SME implementation are described below.

Comparison

Both a component approach and a cognitive approach help one arrive at plausible atoms for the creation of comprehensive, interoperable design software. One expects that the kinds of software that would result from the two approaches would be very different and could thus lead to very different scenarios of future architectural practice. Choice of a research and development approach should be guided by a projection of the implications of the alternative approaches.

A major stumbling block in the component-based approach is the necessity of achieving broad agreement upon the definition and classification of thousands or millions of components. New products that can serve as components in a building are always being invented and would need to be added to the catalogue before they could be used in the system. Sophisticated retrieval tools would be necessary to search the catalogue to find a component that meets the particular needs of the design situation. The cognitive approach used in my research has sidestepped a requirement for such a catalogue. A catalogue of functions and behaviors would still be required. However, it might be smaller than a catalogue of components. The concept of interpretations further reduces the size of the needed catalogue by encapsulating many functions and behaviors into coherent issue-based units and hiding the functions and behaviors from the end-user.

The component approach also runs a risk of requiring too much commitment too soon in the design process. When inserting a component, a designer may be declaring attributes by default without carefully considering them. For example, a designer may wish to state that a wall is to be masonry but may be reluctant to choose between concrete block and brick. The cognitive model described above accounts for the incremental addition and refinement of information through adding interpretations of the design solution.

A third limitation of the component approach is that a component library can never be large enough to accommodate innovative design. Almost by definition, innovative design will use existing components in new ways or invent new components and materials. The research that uses a component approach has not successfully addressed how an end-user such as an architect or engineer or building owner could invent new components or find new ways to use existing components. In contrast, the cognitive approach described above explicitly accounts for the invention of new components. A designer may combine forms, functions and behaviors in a new way. Some kinds of innovation may still fall outside the boundaries of the cognitive model described in this paper.

For these reasons, the cognitive model described in this section is attractive for further development. The first question in developing it further was simply "Is this feasible?"

SME IMPLEMENTATION

The Semantic Modeling Extension (SME) is intended to test whether the model of the design evaluation process that is described in the preceding section can be implemented as software and whether that software can be used by designers in solving a design problem.

System architecture

The SME prototype couples a conventional CAD system, AutoCAD(tm), with knowledge-based evaluation tools. Simple routines for extracting data from the CAD model and exchanging it with the evaluation modules were written in AutoLISP(tm) and C. Four evaluation modules were written in Kappa(tm) using object-oriented and rule-based reasoning techniques. Core functionality allows a user to dynamically add and remove evaluation tools, interpret entities in the CAD model and initiate design evaluations. The core functionality was also written with Kappa. The system architecture is shown in Figure 2.

Base classes of the SME core

Seven base classes provide the system interface to all of the evaluation modules in SME. They are reifications of the concepts used in the model of design evaluation that was described earlier.

- *Form.* In SME, this class is primarily a wrapper around an AutoCAD entity. It provides an interface between the AutoCAD entity and the rest of SME to support queries from SME about geometry. It also provides a representation of the material of the geometric or graphic object.
- *Function.* Requirements on the design solution are represented as distinct Function objects. Each Function class provides an assess! method that must determine whether the requirement has been satisfied by the design solution. The assess! methods employ one or more Behavior objects in that determination.
- *Behavior.* The performance of the design solution is represented by Behavior objects. The Behavior classes must provide a predict! method that is responsible for generating a value for the Behavior. In most cases, the value is generated by deriving quantities from the Form objects.
- *Feature.* A Feature object bundles particular Functions with a particular Form from within a particular Interpretation. When the user interprets a CAD entity, a Feature instance is created. The Feature knows what Functions are implied by its creation.
- *Interpretation.* An evaluation issue, such as egress, construction cost, spatial requirements or energy consumption, is represented in SME by an Interpretation object. The Interpretation object manages the list of Feature instances that are produced by interpreting the design. It provides an interpret! method that presents a list of Feature classes from which the user may pick. The Interpretation object must also provide an evaluate! method that initiates the prediction of Behavior values and the assessment of Function satisfaction.
- *Interpretation Manager.* SME employs an Interpretation Manager to allow the user to load and unload modules and focus attention upon a single Interpretation at a time.
- *Virtual Component.* The Virtual Component class provides for interoperability among the evaluation modules. It assures that Form instances, Function instances and Behavior instances are shared to the highest degree that is possible.

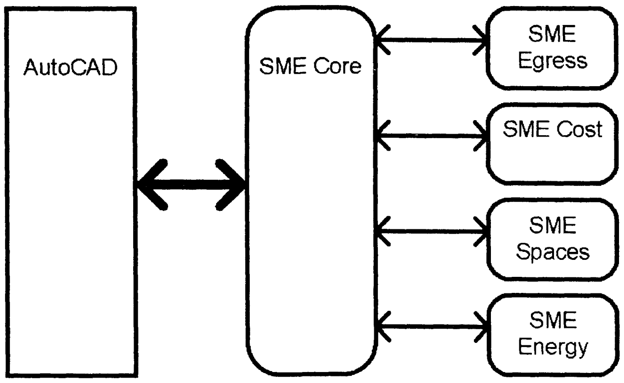

Figure 2, Semantic Modeling Extension system architecture. The components shown with rounded rectangles were programmed with Kappa. The large arrow symbolizes that the link between these components is instituted permanently for a session with SME. The small arrows indicate links that may dynamically be created or severed during a session to accommodate changing needs for evaluations.

Evaluation modules

Each evaluation tool is written using the base classes and is encapsulated as an instance of a subclass of the Interpretation class. An Interpretation Manager object in the SME core is responsible for loading and unloading evaluation modules and also dispatching high level commands to the individual Interpretation instances.

The evaluation tools are implemented to employ a common user interface. They share common dialog boxes for creating features, interpreting CAD entities and inspecting the results of an evaluation.

The reasoning behind the evaluations is implemented in the four evaluation modules using prediction of Behaviors and assessment of Functions. Having set a current Interpretation with the Interpretation Manager, the user may issue a command from the core interface to interpret a CAD entity. The current Interpretation provides the user with a list of Feature classes. When the user selects a Feature class, the system instantiates Functions that are relevant to the particular Interpretation. When the user initiates an evaluation, the SME core dispatches a message to the Interpretation instances to evaluate the design. Each Interpretation sends messages to its Functions to assess themselves. The Functions each ask Behavior instances to predict themselves to provide the information needed for assessment. The results of all of the Function assessments are collected by the Interpretation instances for reporting to the user.

Writing the code for the four evaluation tools was relatively easy. Using object-oriented programming, it was straightforward to reuse code from higher in the class hierarchies. The basic messages defined by the core classes were sufficient for achieving system interfaces for all of the evaluation modules. More detail of the implementation is provided in other papers (Clayton, Kunz, Fischer and Teicholz 1994; Clayton, Kunz and Fischer 1996). The implementation of the evaluation tools is evidence that the cognitive model is correct, useful in understanding evaluation and useful in constructing software to support design.

Interoperability using Virtual Components

In the initial versions of SME, the dominant information shared among the evaluation modules was the CAD graphic representation, which corresponds most clearly to the form of the design. However, the use of Function and Behavior classes suggested an opportunity to share the information expressed in these objects as well. The concept of Virtual Components was developed to provide a mechanism for interoperability.

When the user interprets a CAD entity for the first time, the SME core instantiates a Virtual Component. Each interpreted entity has one and only one Virtual Component instance. The Virtual Component acts as a broker for Function and Behavior instances. As the same CAD entity may be interpreted in different evaluation modules, the Virtual Component collects references to Functions and Behaviors and share them among the evaluation tools. For example, a design object that represents a room may be interpreted as a Laboratory feature in a spatial evaluation module and as a Hospital Room in an egress evaluation module. Both features set minimums for the area of the room, represented as Functions. A shared Behavior that expresses floor area is used to assess the two Functions.

A Virtual Component has no predefined Form, Function or Behavior. The Form, Function and Behavior are incrementally collected during the user's actions of drawing the design, interpreting it, and evaluating it. The collection of the Form, Function and Behavior is performed by the system outside of the attention of the user, who may focus entirely upon the design process. Once all evaluations of the design object are completed, the Virtual Components fully specify the design objects. Because of this ability to collect a design specification at run-time from the actions of the designer in evaluating the design, Virtual Components are particularly suited for design support software directed toward the early stages of design and toward innovative design.

The operations provided by SME for interpreting the design, predicting behaviors, assessing functions and collecting form, function and behavior into Virtual Components appear to be extensible beyond the four evaluation modules and the limited design problem addressed by the SME prototype. Without using a typical product modeling classification of components, SME collects information into a comprehensive representation of a building. Consequently, I refer to building model in SME as a *virtual product model.*

SME TESTS

Usability tests of the SME implementation have demonstrated that the software is easy to learn, easy to use and that it has clear advantages over manual methods for accomplishing design tasks. The test is described in more detail in another publication (Clayton, Fischer, Teicholz and Kunz 1996).

Outline of tests

The software test employed techniques derived from software usability testing. A building design problem was devised that could be solved using the SME prototype or using manual methods. A within-subjects series of trials was used in which some participants first used the computer techniques and some users first used the manual techniques. The participants received about one hour of training for each technique. A time limit of two hours per trial limited the commitment required of participants and placed a time pressure upon the participants. Measurements were taken of time expended to reach various milestones and of accuracy.

Results

The results show that:

1. Participants learned the software quickly, allowing them to complete the trials with only a small amount of training.
2. The manual method and the software supported method employed a similar sequence of actions derived from the cognitive model. The participants accepted the sequence of actions and used it effectively with either the manual method or the computer method.
3. The manual method allowed somewhat more rapid performance. This may partly be due to limited robustness of the software and greater familiarity among the participants with the manual methods.
4. The software-supported method was significantly more accurate than the manual method.

The number of participants was small (only five) and only narrowly representative of design professionals. However, the trials suggest that software that is based on the cognitive model could improve designers' performance, especially in the area of improved accuracy. They also provide evidence that the cognitive model is reasonably accurate in defining the evaluation design process.

CONCLUSIONS

The SME prototype clearly demonstrates that CAD software can be developed based upon a cognitive model of the design evaluation process. It also shows that interoperability may be achieved that shares forms, functions and behaviors rather than components. The trials with the software suggest that designers can quickly learn to use such software and can perform competitively with the software to accomplish design tasks. The experience suggests that a common user interface based upon a formalized design method may be an important part of future comprehensive, interoperable CAD systems.

A theoretical argument suggests that comprehensive, interoperable CAD software that is based upon the cognitive model of interpreting, predicting and assessing could have advantages over software that is based upon component representations. Software that employs "design process-based interoperability" may provide more flexibility and support for creative design than software that employs "component-based interoperability."

REFERENCES

Asimow, M. 1962. *Introduction to Design*. Englewood Cliffs, NJ: Prentice-Hall, Inc.

M.J. Clayton, J.C. Kunz, M.A. Fischer, and P. Teicholz. 1994. First drawings, then semantics. In *Reconnecting, ACADIA 94 Proceedings*, ed. A. C. Harfmann and M. Fraser. Association for Computer Aided Design in Architecture.

Clayton, M.J., M.A. Fischer, J.C. Kunz, R. Fruchter. 1995. Behavior follows form follows function: A theory of design evaluation. *American Society of Civil Engineering Second Congress on Computing in Civil Engineering Proceedings*, 310-317. New York.

Clayton, M.J., J.C. Kunz and M.A. Fischer. 1996. Rapid conceptual design evaluation using a virtual product model. *Engineering Applications of Artificial Intelligence* 9 (4): 439-451.

Clayton, M.J., M.A. Fischer, P. Teicholz and J. Kunz. 1996. The Charrette Testing Method for CAD Research , *ARCC Spring 1996 Conference on Applied Research in Architecture and Planning Proceedings*, Architectural Research Centers Consortium.

Eastman, C. M. 1992. Modeling of buildings: evolution and concepts. *Automation in Construction* 1:99-109.

Gero, J. S. 1990. Design prototypes: A knowledge representation schema for design. *AI Magazine* 11(4): 26-36.

Hansen, K. L., K. D. Johnson and C. B. Tatum. 1990. *Incentives for Integrated Facilities Engineering in the Architectural, Engineering, & Construction Industry*. Center for Integrated Facility Engineering Technical Report No. 37. Stanford, CA.

Harfmann, A. C., and S. S. Chen. 1990. Building representation within a component based paradigm. In *From Research to Practice, ACADIA 90 Proceedings*, ed. J. P. Jordan. Association for Computer-Aided Design in Architecture.

ISO. 1993. *TC184 Part 1: Overview and fundamental principles. Industrial automation systems and integration - Product data representation and exchange*. Draft International Standard, ISO, Geneva, Switzerland, ISO DIS 10303-1.

Industry Alliance for Interoperability. 1996. *IFC Project Model Specifications, version 0.94*.

Gardner, H. 1985. *The mind's new science: A history of the cognitive revolution*. New York: Basic Books, Inc.