Carnegie Mellon

Issues in Modeling and Processing Design Standards

A Paper Presented at the
1992 Computers and Building Standards Workshop
University of Montreal, Montreal, Canada
May 12, 1992

M. Maher Hakim and James H. Garrett, Jr.

Department of Civil Engineering
Carnegie Institute of Technology
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213–3890
(412) 268–5674

ABSTRACT

In almost all engineering design contexts, design standards are used to specify performance requirements. Design standards, or codes of accepted practice, have traditionally existed only in textual form. The complexity of the information expressed within a standard, and the fact that related information is usually scattered over many different sections of a standard, makes standards hard to understand, prone to errors of omission during usage, and subject to multiple interpretations. This problem is compounded by the fact that standards are also subject to change as research leads to improved understanding of behavior.

This paper first discusses the current approaches for developing and using a design standard and then identifies several components that are needed to provide effective computer–based assistance to standards authoring, promulgation and evaluation. Next, the issues that must be addressed in providing these computer–aided standards processing components are discussed. As these issues are raised, an object–oriented approach to addressing these issues is also discussed.

INTRODUCTION

Design standards are developed by committees of researchers and practitioners, for a specific set of products, over many years, and with much iteration. The primary product of such committees has been a *text* describing limits of behavior or form of a particular entity as defined by the standards writing committee. These documents are usually highly cross–referenced, refer heavily to research literature, specify requirements on many classes and subclasses of entities, and contain a large number of evaluation methods that are applicable for different entities and design contexts. In addition, the laborious process of authoring and promulgating a standard limits the frequency at which they are updated and re–issued even though the phenomenological research is continuous. While researchers have developed organizational models of the requirements in a standard and semantic models of the logic used to evaluate each requirement and data item in the standard, little computer assistance, other than generic word processors and possibly some analysis procedures, are currently used for authoring and editing most standards.

The process of standards conformance evaluation has also largely been ignored by those developing computer–aided engineering (CAE) software for design assistance. Their primary focus has been the development of design environments that assist in the synthesis of design solutions, not in the performance of a complete and consistent evaluation of a design for conformance to all aspects of a design standard. In fact, in many CAE systems, it is impossible to describe a completed design and ask the system to check that design for conformance to its representation of the standard. This is because the representation of the standard is in the form of design procedures.

Although some CAE software developers do make the claim that their software will produce designs that "conform to XYZ's standard," most software developers avoid making this claim because doing so opens the software developer up to an enormous amount of additional liability. Only the standards organizations themselves are prepared to take on the responsibility of ensuring that a representation of their design standard is complete, correct, and consistent. For example, AISC has recently announced the ELRFD, an electronic, evaluatable version of their LRFD Steel Design Specification (ELRFD 1990). This particular piece of software had to go through many iterations of committee review and external review before it was released.

The current approach being taken by many CAE software developers in building these design environments is: (1) the applicable portions of these codes and standards are selected by the software

developer (who is usually a junior engineer with little or no experience with that particular standard), (2) these same portions are then interpreted by the software tool developer (with much effort), and then (3) these interpreted standards provisions are implemented as procedural code within the design environment. The fact that it is the software developers, not the standards writing bodies, that are selecting specific parts of the standard and interpreting these code provisions is quite disconcerting, or should be, to the engineering community that uses these design environments. In essence, there is no "third party" standard conformance evaluation of the resulting design and the designer has no access to the representation of the standard that is embodied within the design environment. Thus, even if a designer uses a piece of software that properly claims to produce designs that "conform to XYZ's design specification," he or she has no recourse other than to manually check the completed design for complete conformance to XYZ's standard. With the announcement of ELRFD, the designer now has an alternative to this manual checking of the textual version of the LRFD standard. However, ELRFD is designed as a stand–alone representation of the LRFD standard and requires the designer to manually classify the structural element being checked and manually provide the design data required to check that element for conformance to the LRFD standard.

It is interesting to note that one of the largest "consumers" of construction product, the U.S. Army, has an extremely large collection of in–house standards that have been developed by the U.S. Army Corps of Engineers (USACE). However, most of the design needed by the Army is done external to USACE and hence the USACE spends a lot of time performing standard compliance evaluation. This discussion clearly points to a need for an automated environment in which standards could be authored, delivered, used, and modified.

COMPONENTS OF AN AUTOMATED STANDARDS PROCESSING ENVIRONMENT

From the previous discussion of the current situation in standards authoring and conformance evaluation, it should be obvious that computer assistance in standards authoring should be more than just desktop publishing support. Both the authors and the users of standards will benefit greatly from: (1) a more expressive and computer–evaluatable media than that of a textbook for representing design standards; (2) environments to assist in the development of this representation; and (3) environments to assist in using this representation for either design conformance checking or design synthesis.

An automated standards processing environment would thus require the following four components as shown in Figure 1.: (1) a hypermedia model of the standard; (2) a semantic model of the standard for modeling its semantic content; (3) a distributed authoring environment for developing the hypermedia and the semantic models of the standard by the group of researchers knowledgeable about the standard; (4) a standards processing system. The standards processing system acts as an interface to design product models for which the applicable design standard requirements will be defined, and from which the data needed to evaluate the logic model of these requirements will be retrieved. The standards processing environment will also act as an interface between the CAE environment and the formal model of the design standards. The next four sections discuss the issues to be considered in developing each of these components in an automated standards processing environment.

ISSUES ASSOCIATED WITH THE HYPERMEDIA MODEL OF A STANDARD

A hypermedia–based model of a standard would contain the following: (1) sections of interrelated text stating requirements and definitions to appear in the standard; (2) images and videos of failure

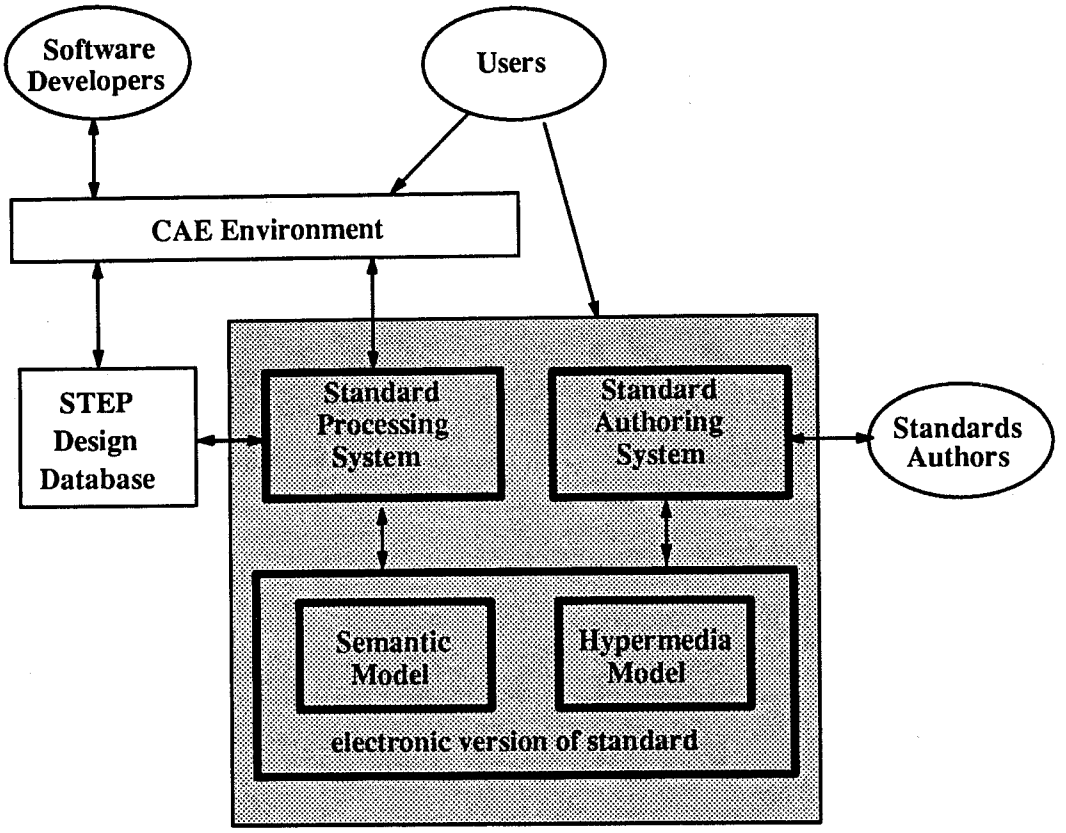


Figure 1. — Automated Standards Processing Environment

modes (and other information benefiting from visual presentation); (3) video lectures (or textual commentary) by those researchers (or their agents) who authored the particular requirement within the standard or who discovered the particular failure mode necessitating that requirement; (4) reference and access to relevant research articles, etc.

One of the primary issues concerning the development of this hypermedia model is that it must also be possible to access the underlying semantic model of this standard and perform conformance evaluations for specific, user–defined design contexts (i.e., perform a standard conformance evaluation exercise using the semantic model of the design standard). Likewise, it should be possible to go from the underlying semantic model of the standard back to the hypermedia representation to get further explanation of the standard.

ISSUES IN SEMANTIC MODELING OF STANDARDS

A provision is a piece of logic within a standard which has the function of assigning a value to a data item. A standard can be viewed as a collection of provisions. Research in standards modeling has focused on two important issues: classification and representation of provisions. The goal of research in classification is to develop classification systems that help designers identify the requirements in the standards that apply to the problem which they are trying to address. The goal of research in representation is to represent those requirements (along with all the provisions which are needed to evaluate the requirements) in evaluatable forms so that they can be checked for confor-

mance with the standards. Much of the research in standards processing has given priority to the problem of representing provisions as opposed to the problem of provision classification. The parsimonious classification system presented in (Vanier 1991) and the approach to standards modeling discussed in (Hosking et al., 1991) are examples of recent efforts which gave priority to the classification problem rather than the representation problem.

The *Classification of Provisions* Problem

The first problem in standards processing, referred to as "the classification problem", involves classifying the requirements and other provisions such that the applicable requirements and provisions for a specific design context can be determined. One approach to solving this problem in the classification tress used in the SASE model (Fenves et al., 1987). Classification trees (or classification networks) serve as indexes (i.e. entry points) to the requirements of the standards. Figure 2 shows a skeletal architecture of the SASE model (Fenves et al., 1987). In this model, each requirement within the standard is associated with one or more leaves of the classification tree. The classification tree is only used to classify the top–level data items of the standard, i.e., the requirements[1]. It is only possible to associate requirements with the leaves of the classification tree; therefore, a link between a requirement and an intermediate node within the classification tree has to be established between this requirement and every child of the that intermediate node.

Another approach to this classification problem, which differs from the SASE approach, is one in which the classification network is established by merging multiple concept class hierarchies each of which addresses one facet of classification[2]. Furthermore, the requirements, as well as all other data items, are associated with one or more nodes in the concept lattice that results from merging the concept hierarchies.

This section discusses the importance of the contextual concept lattice for standards representation and illustrates why the focus in modeling a standard should be on the concept lattice, which represents the context in which each of the provisions of the standard is applicable, rather than on the representation of the those provisions.

The contextual concept lattice consists of several hierarchies of classes that are merged together to form a class lattice. Each concept hierarchy addresses a facet of classification within the standard, such as design component behavior, component shape, material, etc. Figure 3 depicts a graphical representation of the contextual concept lattice. In this figure, a box represents a contextual concept and a circle represents a data item. The figure shows how a contextual concept lattice is formed by merging two concept hierarchies. All data items that have been defined in a concept superclass are inherited by its concept subclasses. The figure illustrates how complex data–item networks in lower–level concepts are formed from merging simpler data–item networks in higher–level concepts and introducing additional data items which are applicable only for those lower–level concepts.

As an example of the contextual concept lattice, consider the Uniform Building Code. In the first four parts of this code, two concept hierarchies can be identified. The first hierarchy represents a classification of buildings based on occupancy, whereas the second hierarchy represents a classification of buildings based on the type of construction. The Uniform Building Code includes: 1) provisions that apply to buildings of different occupancy types regardless of their types of construc-

1. Howard and Fenves considered the problem of classifying data items in (Howard 19?), but still only treated classifiers as identifiers, not as object classes.
2. Harris and Wright introduced the concept of facets of classification in (Harris and Wright 1980), but the individual classifiers were also simple identifiers, not objects.

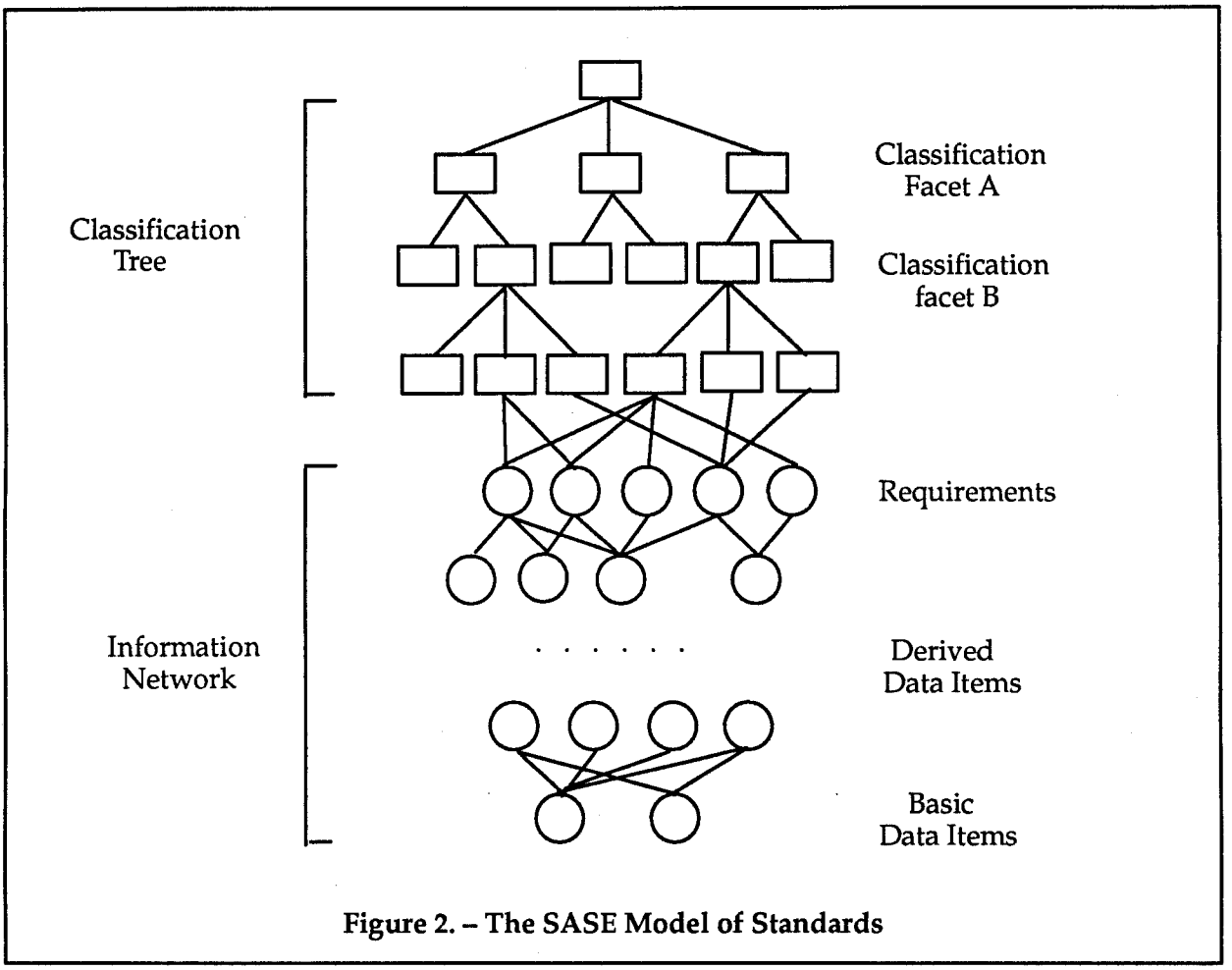
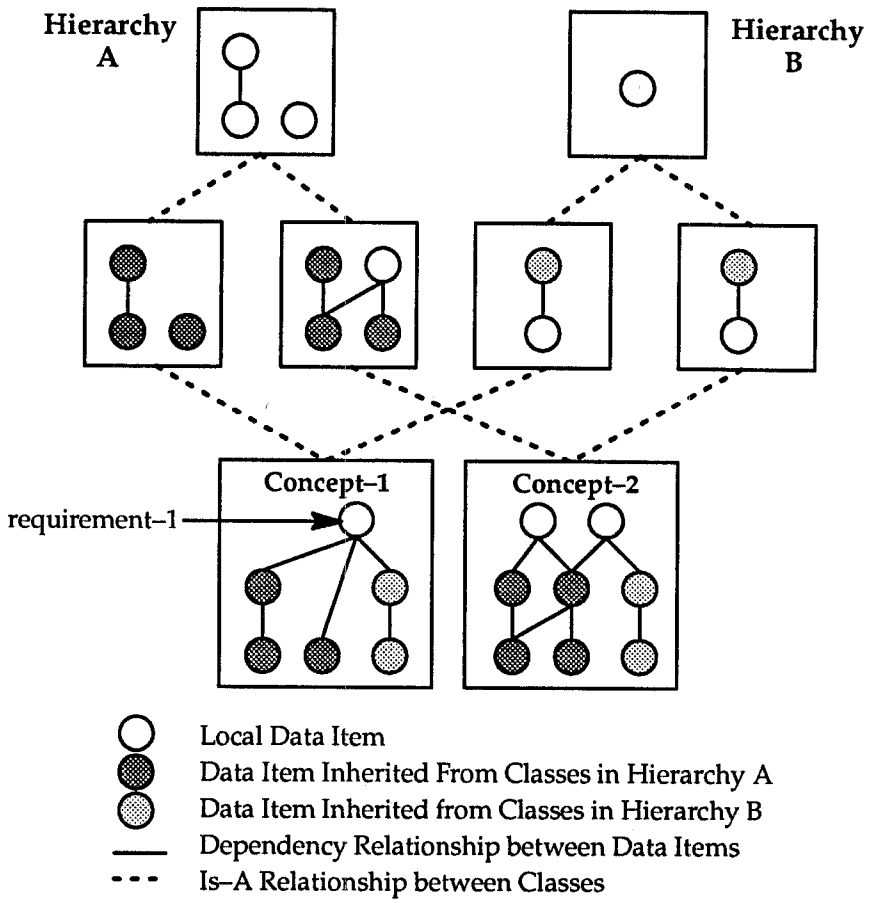


Figure 2. – The SASE Model of Standards

tions, 2) provisions that apply to buildings of different types of construction regardless of their occupancy types, and 3) provisions that apply only to buildings of a specific type of construction and type of occupancy. Provisions which fall under the first category can be encapsulated within the concept classes of the *occupancy* concept hierarchy; provisions which fall under the second category can be encapsulated within the *type–of–construction* concept hierarchy; and provisions which fall under the third category can be encapsulated within the classes of the concept lattice resulting from merging the two concept hierarchies. For example, the natural light requirements depend only on the occupancy classification and, therefore, can be encapsulated within the classes of the occupancy concept hierarchy; the stair construction requirements depend only on the type of construction and, therefore, can be encapsulated within the classes of the type–of–construction hierarchy; and the allowable floor area requirement and maximum height requirement depend on both the occupancy type and type of construction and, therefore, can be encapsulated in the concept classes which result from merging the two hierarchies mentioned above to form a contextual concept class lattice. Figure 4 shows a partial contextual concept lattice extracted from the Uniform Building Code. The figure also shows examples of the requirements that can be embedded within the classes of the concept lattice.

The contextual concept lattice serves three purposes within the model of a standard. First, it identifies the classes of concepts to which the standard is intended to apply. An instance of a contextual concept inherits all applicable standard information from its parent class. Second, it organizes and



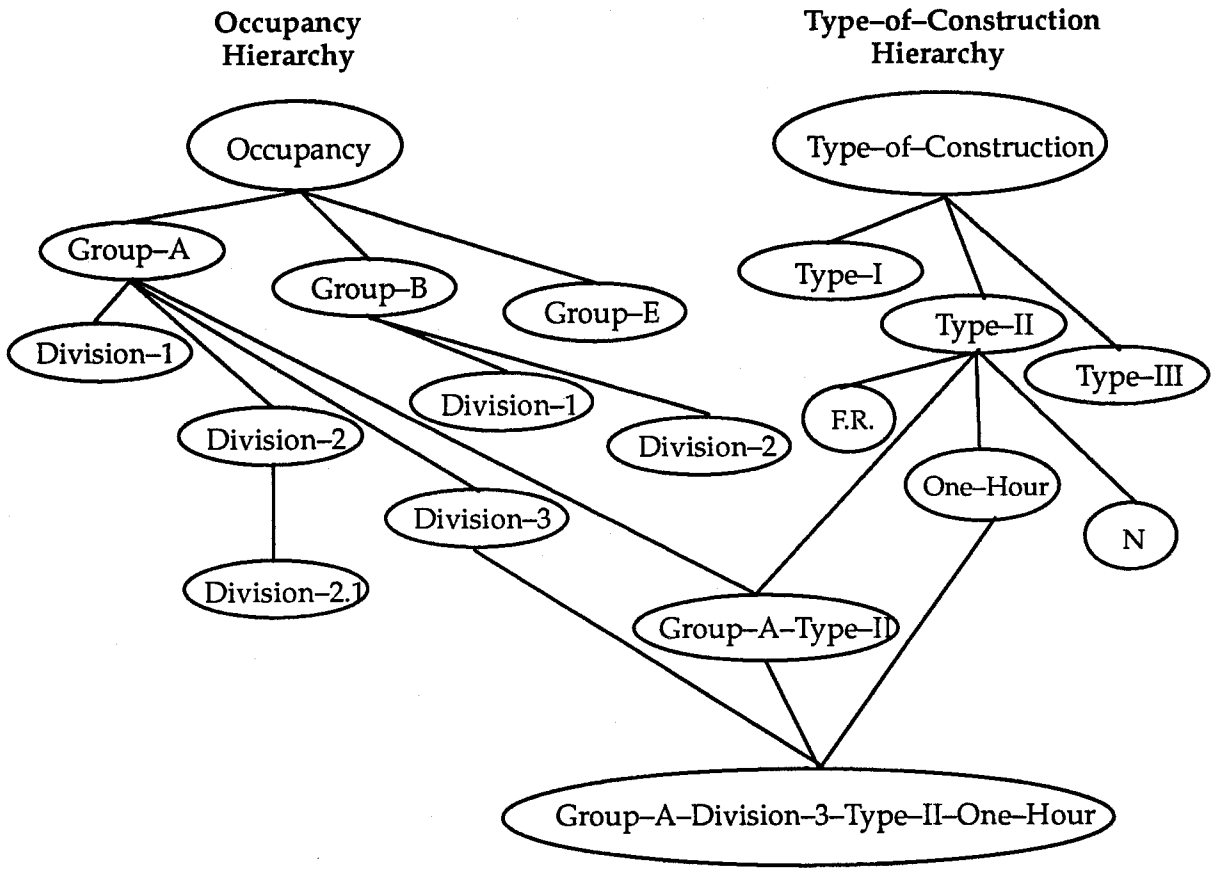
Notice that *requirement–1* applies only in the context when the design object can be identified as an instance of the *concept–1* class. The *requirement–1* data item is dependent on other data items, which are inherited form concept classes defined in hierarchies A and B.

Figure 3. – The Object–Oriented Model of Standards

differentiates between the various specific methods for evaluating derived data items defined within a standard. Third, it defines the context (through the merging of concepts) within which the provisions of the standard can be embedded, and hence serves as a mechanism by which the model of the standard can be mapped to the data of an external product model.

Four advantages are gained by using the contextual concept lattice to classify the provisions of standards:

- The classification network proposed in the SASE model is used only to index and access the provisions, not to define a context within which the provisions can be embedded. An application that uses the SASE model must, therefore, explicitly specify the classifiers that identify the applicable provisions for each context. For example, in order to access the provisions applicable to an *I–shaped–cross–section column*, a design application must explicitly identify the appropriate classifiers (type–of–component is *column*, shape of component is *I*), and use them to access the standards requirements. Using the contextual concept lattice described above, a design object created within a design application can first be identified as an instance of one or more con-



Group–A

natural–light–requirement:
if not *sufficient–artificial–light*
then *exterior–glazed–opening–area* >=
1/10 x *total–floor–area*

ventilation–requirement
if not *sufficient–mechanical–ventilating system*
then *exterior–opening–area* >=
1/20 x *total–floor–area*

Group–B

natural–light–requirement:
if not *sufficient–artificial–light*
then *exterior–glazed–opening–area* >=
1/10 x *total–floor–area*

ventilation–requirement
if not *sufficient–mechanical–ventilating system*
then *exterior–opening–area* >=
1/20 x *total–floor–area*

Group–A–Division–3–Type–II–One–Hour

Allowable–Floor–Area–Requirement:
if *no–of–stories* = 1
then *total–floor–area* <= 13,500

Maximum–Hight–Requirement
hight–of–building <=65
no–of–stories <= 2

Figure 4. – A Partial Contextual Concept Lattice for the Uniform Building Code with a few examples on the requirements embedded in the concept classes

textual concept classes and then can directly access all of the applicable provisions of the standard which are encapsulated within that class. Hence, the need to map context into classifiers that describe the access paths to each of the provisions is avoided. Therefore, the contextual concept lattice allows for better integration of the standard model within an automated environment.

- All provisions of a standard, not only the requirements, are classified within the concept lattice.
- Since the provisions of a standard are embedded within the contextual concepts, every data item within this standard also has to be defined in a certain context, be it general or specific. This forces standard modeler to clearly define the context of every provision of the standard and aids in identifying the incomplete and inconsistent information within the standard.
- The use of multiple inheritance in the contextual concept lattice greatly simplifies the representation and avoids the unnecessary repetition of classifiers and access paths to the same provisions when these provisions can be applied in different contexts. The fact that the classification system in the SASE model is represented as a tree, which has a single root node, forces nested classifications which duplicates declarations of classifiers, branching nodes, and provisions.

The use of multiple inheritance in the contextual concept lattice avoids the unnecessary repetition of classifiers and access paths to the same provisions when these provisions can be applied in different contexts. The use of multiple inheritance is justified for building a computer model of a standard by computer programmers who understand the implications of using multiple inheritance. However, our goal is to develop an approach to standards modeling that facilitates authoring, as well as using, models of standards. In an interactive modeling environment, the standard modeler will first create the concept hierarchies of the standard, then start merging these hierarchies to create the context for the provisions of the standards. There are two related problems associated with using multiple inheritance for representing contextual information: (Refer to Figure 6)

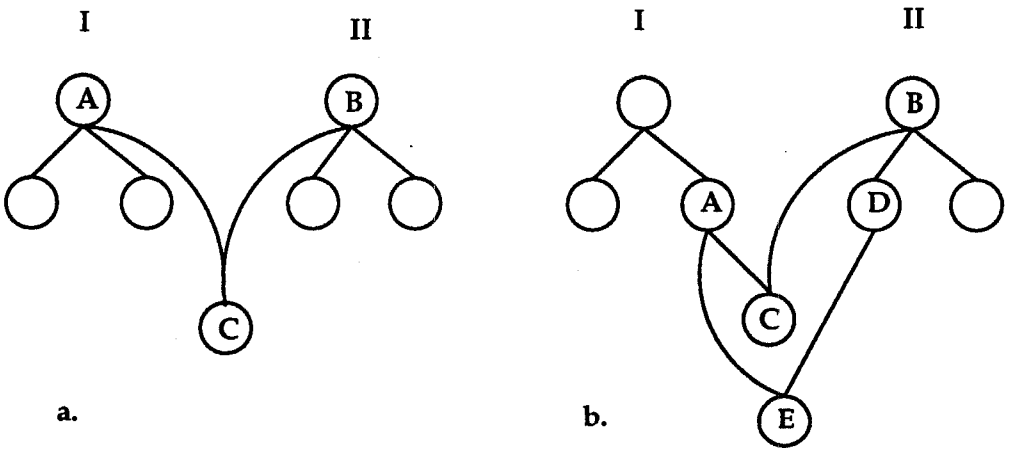


Figure 5. – Problems with creating the contextual concept lattice

- If a provision is to be applied in a context which results from merging two (or more) intermediate–level concepts (as in Figure 6.a), the user has to explicitly specify that the provision also applies in contexts which result from considering all merge combinations of the child nodes of those intermediate nodes. In Figure 6.a, provisions added to Class C are also applicable in all the subclasses resulting from merging A and each of its subclasses with B and each of its subclasses. The user has to manually define all these merge combinations and identify them as sub-

classes of the C concept class. For example, a requirement that is defined to be applicable within the concept Group–A–Type–II is also applicable within the concepts Group–A–Division–2–Type–II, Group–A–Division–3–Type–II, Group–A–Type–II–One–Hour, Group–A–Division–3–Type–II–F.R., etc.

- A user may create a situation, like the one shown in Figure 6.b, where links between classes and subclasses are missing. In Figure 6.b, class E should have been made a subclass of C, because all provisions which apply within the context of C should also be applicable within the context of E. For example, a link between concept classes Group–A–Type–II and Group–A–Division–3–Type–II–One–Hour in Figure 4. has to be created explicitly by the standard modeler.

These problems can be solved by using the subsumption relationship rather than the subtype relationship when creating the contextual concept lattice. A concept is said to subsume another concept if all of its instances at anytime include all of the instances of the other concept (Borgida et al., 1989). The subsumption relationship is usually determined by comparing the classification criteria or "membership conditions" of the concept classes. This topic is the focus of current research conducted by the authors.

The *Representation of Provisions* Problem

Much work has been done on representing the logic of evaluation of the provisions of the standards (traditionally referred to as data items). In principle, a data item can be represented as a function that returns a single value. However, since the evaluation of derived data items often involves the calculation of many intermediate quantities, several declarative representations have been proposed such as decision tables (Fenves, et al. 1987), rules (Rasdorf and Wang 1988), predicates (Rasdorf and Lakmazaheri 1990), and objects (Garrett and Hakim 1992).

As mentioned above, provisions of a standard can be represented as derived data items embedded in the classes of the concept lattice. Much research has been expended on developing representations of the logic for evaluating derived data items. Many researchers have proposed representing the requirements of design standards and design regulations as design constraints, and use constraint propagation mechanisms to manipulate them (Yokoyama 1990, Murtagh and Shimura 1990). Other models used different representation schemes for representing the provisions of standards, such as decision tables (Fenves 1966, Fenves, et al. 1987, Garrett and Fenves 1987, Lopez, et al. 1989), relations in a relational database (Fenves and Rasdorf 1985), first–order predicates (Jain, et al. 1989, Rasdorf and Lakmazaheri 1990), production rules (Rasdorf and Wang 1988), facts (Topping and Kumar 1989), nodes in a hypertext medium (Cornick 1991), and objects in an object–oriented paradigm (Garrett and Hakim 1992).

The issue of representing derived data items, though important, has distracted researchers from focusing on the more important issue of representing the scope of standards, that is, the context within which the provisions of the standard can be applied. Derived data items can be represented using any of the declarative representational constructs described above and then parsed into a function (or, using object–oriented programming terminology, a method). Since all derived data items are represented as methods which call each other whenever necessary, an explicit description of the dependency relationships among the various data items (such as the information network in the SASE model in discussed in Fenves, et al. 1987), or even an implicit one (such as the ingredients slot in (Garrett and Hakim 1992)) is no longer needed. The price paid for representing derived data items as methods is that the evaluation of these data items is always dynamic. No longer is it possible to store the value of a data item and invalidate it only if one of its direct or indirect ingredients value has been changed. However, this price is negligible for the following reason: the conformance checking of a design object for compliance with the requirements of a design standard is usually

carried out only once as long as the basic data items of this object are not changed. When the value of a basic data item is changed, the invalidating algorithm employed in the SASE model and in (Garrett and Hakim 1992) recursively traverses the information network up to the root data items and deletes the values of all the derived data items which are dependent on the affected data item. The time consumed by this algorithm is often larger than the time that would have been spent re-evaluating the requirements which were not affected by the changed value of the basic data item. Furthermore, considering the current status of speed of computation, focusing on minimizing the computations involved in rechecking the conformance of a design object, which is typically on the order of tens, or possibly hundreds of computations, is unjustifiable.

ISSUES ASSOCIATED WITH THE DISTRIBUTED AUTHORING ENVIRONMENT

When the standard is represented using both hypermedia and semantic models, it opens up the possibility of fundamental changes in the standard development process. For example, the group of researchers composing the standard development committee need not meet together physically as a group, but instead can participate in the concurrent development of a standard using a logically centralized database to track versions and alternatives of the standard through the standards development process. By having a hypermedia model of the standard, links between the provisions in the standard and the supporting material and commentary will be naturally developed. Such a representation would also allow for the intent behind suggested provisions or changes in provisions to be much more apparent. By having a semantic model of the standard, it will be possible to perform analyses and evaluations of the standard as it evolves within the committee.

By making the standards development process a concurrent design activity, it will make the development of a logically centralized computer–based representation a natural by–product of this activity. Once this computer–based representation consisting of both a hypermedia model and a semantic model, it can be more easily updated, evaluated for completeness and consistency, and applied to a specific design product model to determine the conformance of that product to the standard. Thus, the combination of a hypermedia environment, semantic model editors, and an additional set of tools for analyzing the completeness and consistency of these models, similar to those in SASE (Fenves 1987), will form a sophisticated and highly useful tool for standards authoring.

ISSUES IN INTERFACING STANDARDS WITH DESIGN PRODUCT MODELS

Computer–aided design environments which support engineering design tasks have recently started to utilize central databases, as opposed to data files, as a means for integrating computer–aided design tools and applications. The database technology developed in recent years provides facilities that are useful for managing the repository of data generated in engineering design applications. The design applications perform computationally intensive tasks such as design analysis, drafting, simulation, standards processing, etc. (See Figure 6).

A design product model built in the central database serves as a global repository for design information that can be shared among design applications. Every design application should provide an interface to the design product model. In an integrated design environment, a design application is started by a designer according to predefined protocols. A standard processor, which is considered as a design application, can be invoked by the designer whenever standard conformance checking is required. The standard processor should provide an interface to the design product model that serves two purposes: 1) extract the design data needed to perform the standard evaluation, and 2) return the evaluation results back to the design product model. Whereas the second

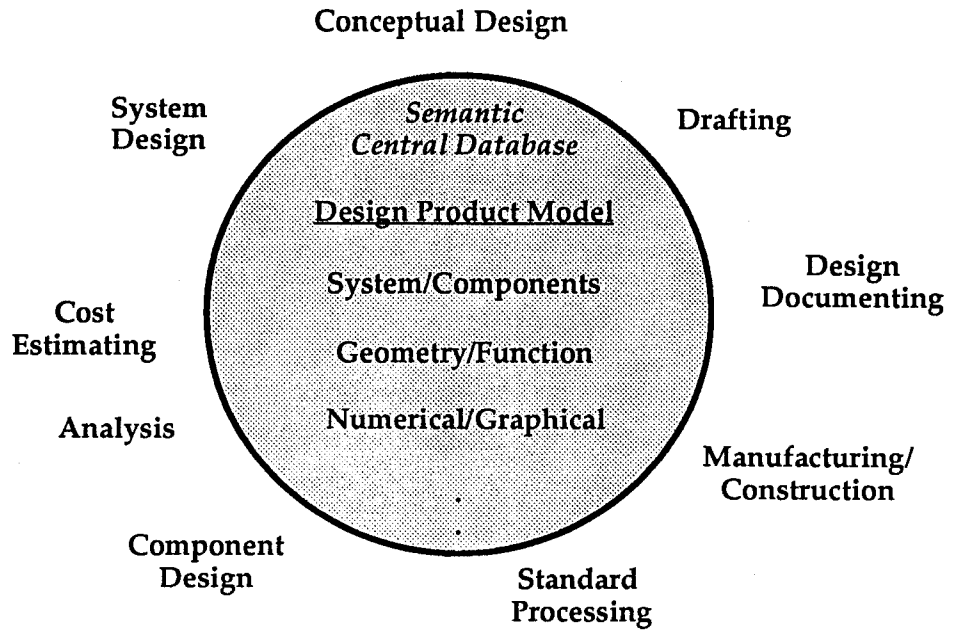


Figure 6. – An Integrated Design Environment

task can be considered to be trivial, the first task is far from trivial. The standards processor should be able to identify and extract two types of design data from the design product model. The first type is the *context identification design data* which enables the standards processor to identify the design context, i.e., identify the contextual design concepts to which the current design situation belongs. This step is an important task of the standards processor that has been ignored in current research on standards modeling. The second type of design information is the *evaluation support design data* which enables the standards processor to evaluate the requirements of the standard which have been identified to be applicable within the current design context. The evaluation support design data are the values for the basic data items for which a standard does not provide a method for evaluation and, thus, have to be provided from an external source to the model of the standard.

To be able to utilize the context identification design data to identify the relevant contextual design concepts, a model of design standards should include classification criteria. The classification criteria represent computable definitions of the concept classes defined within the standard. These criteria enable a standard processor to identify instances of products in a design product model as instances of relevant contextual design concepts within the model of the standard. In some standards, such as the Uniform Building Codes (UCB 1988), classification provisions are stated explicitly in the standard. Classification criteria can also be used to guide in the building of the contextual concept lattice. In other standards, however, classification criteria are not explicitly stated within the standard. For example, the Load and Resistance Factor Design Standard (LRFD 1986) does not include provisions to classify structural members. The classification of a structural member as a column, beam, or beam–column depends on the judgement of the designer. An automated model of a standard that can interface with a design environment has to include a clear, consistent, and unambiguous representation of the classification criteria. The model of a structural design stan-

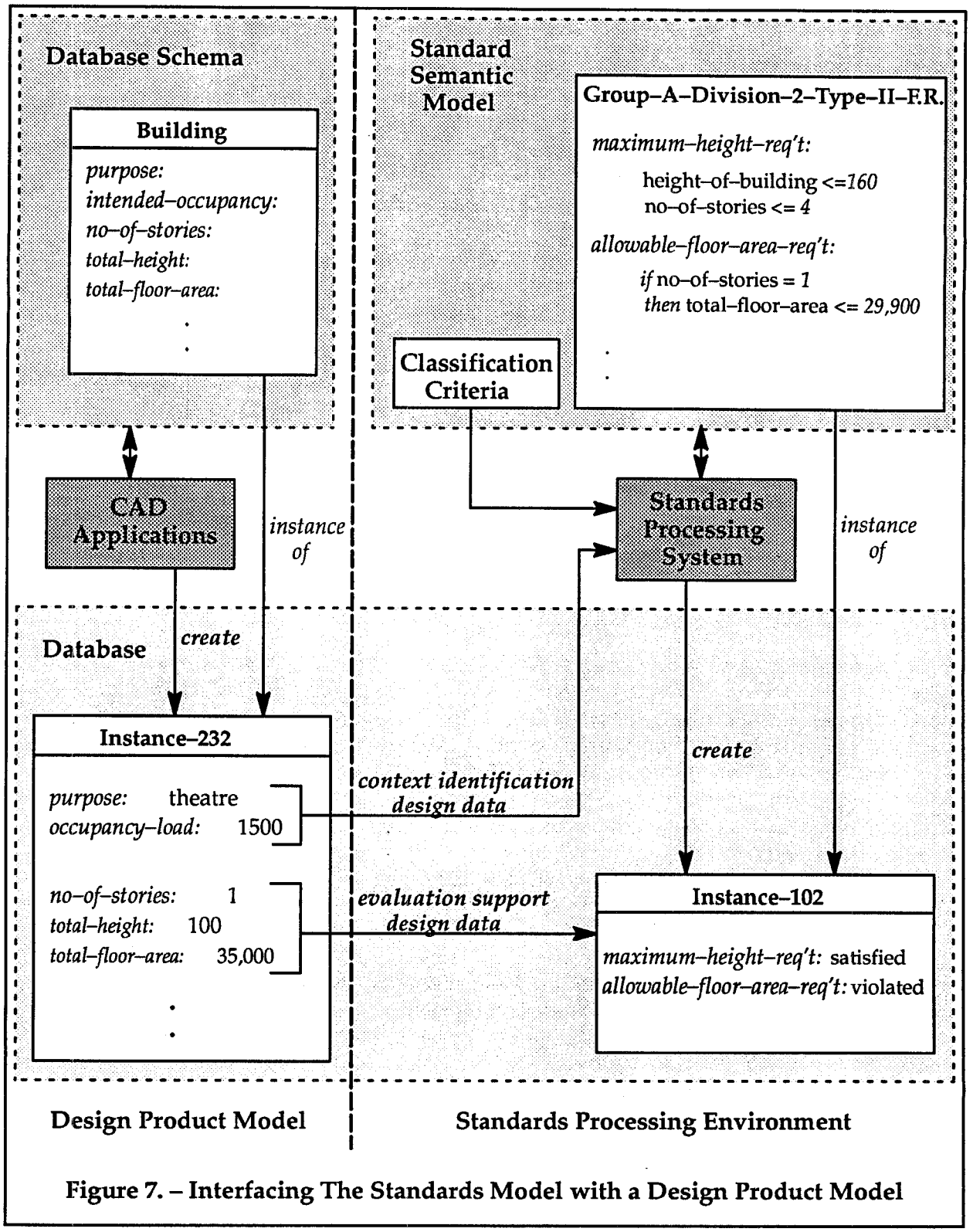


Figure 7. – Interfacing The Standards Model with a Design Product Model

dard (such as LRFD) which interfaces with a design product model of a structural component should be able to classify the component according to the concept hierarchies present within the standards (such as behavior, shape, material properties, etc.). Therefore, heuristic classification criteria which is based on knowledge and experience should be included in the model for such standards. If classification criteria are not included in the model of the standard, designer involvement becomes a necessity for the contextual identification. This involvement, if not minimized, eliminates the guarantees for completeness, consistency and correctness when using an automated standards processing environment.

Consider for example the case of an architectural design of a building. The product model of the building includes a schema definition of the building and all its parts. In this schema, the type *building* might be defined to describe the attributes of the building as a whole (See Figure 7.). Suppose that there is an object of type *building* which has been designed (using various CAD applications) and needs to be checked for conformance with the Uniform Building Code (UBC 1988). The first task of a standards processing system is to try to classify the object according to the UBC's classification criteria. The classification task involves identifying the contextual concept class(es) to which the object should belong. Once the object is identified to be an instance of one or more contextual concept classes, the requirements that are encapsulated within this concept class can be evaluated. The evaluation of these requirements will require retaining information (basic data items) from the object supplied by the product model. In the example shown in Figure 7., the standards processing system classifies the *building* object *instance–232* to be of type *Group–A–Division–2–Type–II–F.R.* The standards processor successfully carries out the classification task because all context identification design data required for the classification task could be provided by the product data. This data includes the purpose of the building and the intended occupant load. For the purpose of evaluating the standard, the standards processing system creates an instance of the identified contextual concept class to which the building belong and link this instance to the building object within the design product model. The standards processor then attempts to evaluate the requirements applicable within the instance object. All data items that can not be evaluated within the concept instance (*instance–102* in Figure 7.) are considered as basic data items, and therefore, are delegated to the product design object (*instance–232* in Figure 7.). The figure shows that the basic data items *total–height* and *total–floor–area* defined in *instance–232* of type building in the design product model, are used to evaluate the requirements *maximum–height–req't* and *allowable–floor–area–req't* in *instance–102* of type *Group–A–Division–2–Type–II–F.R.* in the UBC's semantic model.

ISSUES IN INTERFACING STANDARDS WITH CAE SYSTEMS

The discussion within this paper has predominantly focussed on the issues associated with the modeling of a design standard and the issues associated with a post–design conformance evaluation of such a standard. However, Garrett and Fenves also investigated the usage of standards during the synthesis phase of the design process (Garrett and Fenves 1987, 1989). The main idea was to take a formally modeled design standard and perform the following steps: (1) identify the set of applicable design standard requirements; (2) focus on a subset of those requirements for the purposes of design synthesis; (3) generate a set of design constraints from the formal representation of the design standard requirements; (4) solve that set of constraints; (5) evaluate the resulting design against all of the other applicable design standard requirements that were not in focus (identical to conformance evaluation); and (6) modify the design focus, the constraint set, or the design itself to account for violations of design requirements. Garrett and Fenves clearly showed how a formal, declarative representation of the design standard requirements could be manipulated into a set of constraints and solved to develop a design solution. However, the process of focussing the design

process was a much less well developed concept and remains the primary issue concerning this form of standards usage.

The classification knowledge within the design standard acts as a "mapping" from the design product model into the contextual concept lattice of the standard and can be used to identify all applicable design standard requirements for the modelled product, The knowledge that is used to focus the design process on a particular subset of standard requirements would be used to perform several functions: (1) it would be used to "fill in" those attributes within the product model that are not known, but need to be, in order for the product model to be classified using the above described classification knowledge; and (2) it would be used to "prune" the list of standard requirements found using this previously obtained classification by "looking for" additional aspects within the design context. At this time, it is believed that the first form of focussing knowledge, the product model completion knowledge, will have to be represented as process outside of the model of the standard but still within the standards processing environment; the second form of focussing knowledge, the requirement pruning knowledge, can be represented within the contextual concept lattice along with the classification knowledge.

CLOSURE

In this paper, the major components of a computer–assisted standards processing environment were identified to be: a hypermedia model of the standard, a semantic model of the standard, a distributed environment for authoring these models, and a standards processor for mapping between the design standard models and a model of the products to be checked. Several major issues were identified for each of these components. One of the major issues is to provide the ability to flexibly model the contexts for which the standard provisions are defined and the ability to map between these contextual contexts and the model of the product to be checked. Given these two abilities, the standard will be able to be modeled independently of the product models to which it will be applied. The mapping process will occur at runtime when the standards processor takes the product model to be checked and classifies it as an instance of a set of contextual concepts within the standard. Once this link has been made, the product model inherits all applicable standard provisions and can be checked for conformance.

The standards processing environment described in this paper is currently under development and is being implemented in the common lisp object system (CLOS). While various components of the model (e.g., the semantic model for representing provisions) have been implemented, other components of the integrated system are either yet to be developed (e.g., the hypermedia model and editors) or under development (e.g., the set of editors for creating the semantic model and the standards processor for mapping between the semantic model of the standard and the product model).

ACKNOWLEDGEMENTS

This material is based on work supported by the National Science Foundation under Grant No. DMC–8808132 and is currently supported under Grant No. DDM–8957493. Matching funds have been kindly provided by the Digital Equipment Corporation, the American Institute of Steel Construction, and the Association of American Railways.

REFERENCES

American Institute of Steel Construction, Inc. 1986. *Load and Resistance Factor Design Specification for Structural Steel Buildings*. Chicago, Illinois.

Bordiga, A., R. J. Brachman, D. L. McGuinness, and L. A. Resnick. 1989. CLASSIC: a structural data model for objects. *Proceedings of the ACM–SIGMOD*. Portland, Oregon.

Cornick, S. M. 1991. HyperCode: The building code as a hyperdocument. *Engineering with Computers* 7: 37–46.

ELRFD Checker/Browser User's Guide, 1990. Trechnical Report, Visual Edge Software Ltd., Quebec, Canada.

Fenves, S. J. 1966. Tabular decision logic for structural design. *Journal of Structural Engineering* 92(6): 473–490.

Fenves, S. J., R. N. Wright, F. I. Stahl, and K. A. Reed. 1987. *Introduction to SASE: Standards Analysis, Synthesis and Expression*. Technical Report, NBSIR 873513, National Bureau of Standards, Washington, D.C.

Fenves, S. J. and W. J. Rasdorf. 1985. Treatment of engineering design constraints in a relational database. *Engineering with Computers* 1: 27–37

Garrett, J. H., Jr., and S. J. Fenves. 1987. A Knowledge–based standard processor for structural component design. *Engineering with Computers* 2(4): 219–238.

Garrett, J. H., Jr., and S. J. Fenves. 1989. A Knowledge–based standard–independent member design. *Journal of Structural Engineering*. 115(6): 1396–1411.

Garrett, Jr., J. H., and M. Maher Hakim. 1992. An object–oriented model of engineering design standards. *Journal of Computing in Civil Engineering*. To appear.

Harris, J. R. and R. N. Wright. 1980. *Organization of building standards: systematic techniques for scope and arrangement*. Technical Report, Building Science Series NBS BSS 136, National Bureau of Standards, Washington D. C.

Hosking, J., W. Mugridge, and J. Hamer. 1991. An architecture for code of practice conformance systems. *VTT Symposium 125: Computers and Building Regulations*. Espoo, Finland.

Jain, D., K. H. Law and H. Krawinkler. 1989. On processing standards with predicate calculus. *Proceedings of the Sixth Conference on Computing in Civil Engineering*. ASCE. Atlanta, Georgia.

Lopez, L. A., S. Elam and K. Reed. 1989. Software concept for checking engineering designs for conformance with codes and standards. *Engineering with Computers* 5: 63–78.

Murtagh, N. and M. Shimura. 1990. Parametric engineering design using constraint–based reasoning. *Proceedings of the Eight National Conference on Artificial Intelligence*. AAAI..

Rasdorf, W. J. and T. E. Wang. 1988. Generic design standards processing in an expert system environment. *Journal of Computing in Civil Engineering* 2(1): 68–87.

Rasdorf, W. J. and S. Lakmazaheri 1990. Logic–based approach for modeling organization of design standards. *Journal of Computing in Civil Engineering* 4(2): 102–123.

Topping, B. H. V. and Kumar, B. 1989. Knowledge representation and processing for structural design codes. *Engineering Applications of AI* 2(9): 214–227.

International Conference of Building Officials. 1988. *Uniform Building Code*. Whittier, CA.

Vanier, D. J. 1991. A parsimonious classification system to extract project–specific building codes. *VTT Symposium 125: Computers and Building Regulations*. Espoo, Finland.

Yokoyama, T. 1990. An object–oriented and constraint–based knowledge representation system for design object modeling. *Proceedings of the Sixth Conference on Artificial Intelligence Applications. IEEE*. Santa Barbara, California.