An Integrated System for Design Standards Processing*

Kincho H. Law[1] and Nobuyoshi Yabuki[2]

Abstract

This paper describes an integrated model for the documentation, representation, and processing of design standards in a Computer Aided Design (CAD) environment. We combine object-oriented and logic programming paradigms to provide a unified Object-Logic model to represent and process design standards. In this model, a designer can check the design for compliance with design provisions as well as perform component design. Furthermore, the Object-Logic model also allows checking the properties of completeness and uniqueness of design standards. Besides the Object-Logic representation, the framework includes the storage of background information such as commentaries of the design provisions in a HyperDocument environment. The Object-Logic and HyperDocument models are integrated into a unified framework. To demonstrate the feasibility and practicality of this framework, a prototype system has been implemented for the American Institute Steel Construction (AISC) Load and Resistance Factor Design (LRFD) specification.

Key Words: design standards, object-oriented programming, logic programming, hypertext, HyperFile, conformance checking, component design, standards analysis, document storage and retrieval.

1. Introduction

Design Standards play a significant role in the design process to ensure safety, quality, and functionality of civil engineering structures and facilities. Design standards contain a large amount of complex information that an engineer would need a great deal of experience to comprehend and use the code correctly and effectively. Checking a design for conformance with applicable design codes (conformance checking) and designing using standards is a tedious, laborious, and difficult task. Misinterpreting or overlooking provisions of a design code could have serious consequences.

A common approach for incorporating design standards processing into CAD software is to encode the provisions into a computer program using procedural languages. This approach makes it difficult to modify the program when the standard is revised. This "hard-coding" approach often aims for either conformance checking or automated designing, but it is rather difficult to implement both applications in a unified manner. Furthermore, current conformance checking programs that are available in the industry are mostly stand-alone type. They are usually not integrated with CAD data models, engineering databases, and Computer Aided Design and Drafting (CADD) systems.

During the code developing process, many background documents and data are collected. Although this information gathering process is time-consuming and expensive, the documents

* *To be presented at the CIB W78 & Computers in Building Standards Group (CBSG) meetings in Montreal, Canada, May 12-14, 1992.*

[1]Associate Professor, Department of Civil Engineering, Stanford University, Stanford, CA 94305-4020

[2]Graduate Student, Department of Civil Engineering, Stanford University, Stanford, CA 94305-4020

collected are often not appeared in the code and not shared by other researchers and engineers. This background information could be used to provide explanations for design applications and served to clarify the provisions for code development purpose.

For more than two decades, many researchers and engineers have attempted to computerize design standards for design application and conformance checking purposes. The recent developments in artificial intelligence and hypertext technologies could have significant impact on the representation of design codes, the processing of design standards, and the integration of design standard processing in an intelligent Computer Aided Design (CAD) environment. The objective of our research is to develop a model that can perform both conformance checking and component design within the same environment as well as checking the completeness and consistency of design standards by representing both the organization and provisions of the design standards effectively. In addition, the model should be able to store background documents and knowledge that can be accessed by code writers and engineers, and should be integrated in a CAD environment.

There are two distinct categories of knowledge in a design standard:

- knowledge of the organization of design objects, and
- knowledge of the methods in reasoning about design.

The object-oriented paradigm lends itself naturally to representing the organizational aspect of the design standard. The logic programming paradigm, on the other hand, is well suited to implementing the reasoning mechanisms for design. We combine the object-oriented and logic programming paradigms to provide a unified Object-Logic model for the representation and processing of design standards. By storing the design provisions in a "fact" base, the model is capable of performing conformance checking and component design, and syntactically analyzing the standard. Besides the Object-Logic representation, we propose a framework to include the storage of background information (such as commentaries) of the design provisions in a HyperDocument model, which is based on a form of HyperFile structure [Clifton 90]. The HyperDocument model allows storage and access of heterogeneous documents, including design provisions, their background information and data, and programs of a design standard. The HyperDocument system also allows designers and engineers to obtain explanations and other background information to support design tasks. Furthermore, by comparing the design provisions and the corresponding program codes, semantic correctness of the provisions can be checked.

The Object-Logic and the HyperDocument models are integrated into a unified Hyper-Object-Logic model. The overall architecture of the Hyper-Object-Logic model is depicted as shown in Figure 1. The Object-Logic and HyperDocument models are integrated together by sharing Method Objects, which are program codes representing the design provisions, and are stored in the Standards Base. This integrated system for the representation, processing, and the documentation of design standards is the subject of this paper [Yabuki 92].

2. Computer Representation of Design Standards

In our prototype system, the Object-Logic model consists of five basic modules:

- CAD Object Data Base, which facilitates member definition using the Object Model and retrieving member data from engineering databases.
- Standards Base, which represents the organization and provisions of the standard,
- Conformance Checking Module, which performs conformance checking of a given member,
- Component Design Module, which generates component design of a given member, and

- Standards Analysis Module, which checks completeness and uniqueness of provisions and the organization of the standard, and also examines the relations among the provisions.

These modules are briefly described in the following subsections.

2.1 CAD Object Data Base

A design member object such as a beam or a column consists of attributes, or properties of the member, and external constraints given by the user. To define the attribute values and external constraints to the design member, a CAD Object Data Base (Figure 2) is provided in the Object-Logic model. The main components of the CAD Object Data Base are

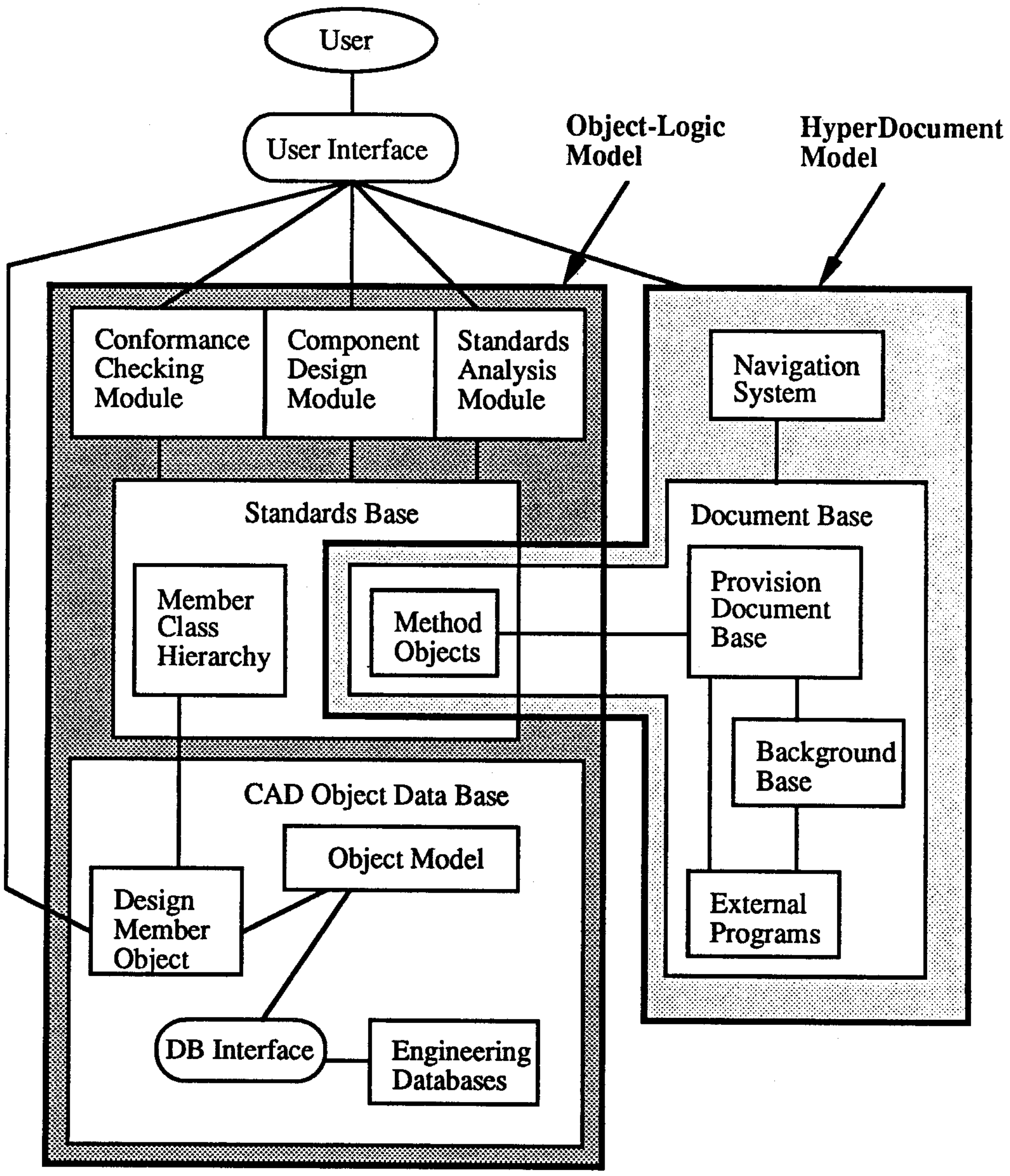


Figure 1 Overview of the Hyper-Object-Logic Model

- an Object Model, which is a hierarchical structure representing the member objects, and
- engineering databases, which contain data that is not often included in the standard such as the dimensions of component shapes and properties of various materials.

In a CAD environment, the attribute information about the design member object may be stored in a relational or object-oriented database system. If an attribute value of the design member object is not given, the data is retrieved from an engineering database. Since relational calculus is a subset of predicate logic, the uniformity of the relational database and Object-Logic model can be retained. The Object-Logic model also provides a means to integrate the design system with a CAD object database system with an appropriate interface.

As an example, let us define a steel column "column_25" as shown in Figure 3. We define "column_25" as an instance of a class "steel column" in the Object Model as shown in Figure 2. "Column_25" inherits attributes from the class "steel column." Once the member "column_25" is defined, we assign attribute values and external constraints (if any) for the design member. Since "column_25" is a wide flange W shape, the design object is linked to a Data Object "wshape" as an instance to retrieve attribute data that are not given by the user from the engineering database "WSHAPE_DIM."

2.2 Standards Base

The Standards Base consists a Member Class Hierarchy, (representing the organization of the standard), and Method Objects, (corresponding to the provisions of the standard). The Member Class Hierarchy is an object-oriented organizational model of design standards, which is partially based on a frame-based classification system proposed by Garrett and Fenves [Garrett 86]. The Member Class Hierarchy consists of classes, which contain attributes and pointers to Method Objects. Figure 4 shows an example of the Member Class Hierarchy. Given a design member, the Member Class Hierarchy can be used to identify all the applicable provisions by traversing the hierarchical tree based on the attributes and properties of the member. The Member Class Hierarchy includes both "AND" and "OR" relations. When traversing the AND-OR Member Class Hierarchy, all nodes under an AND node are traversed, while only one node under an OR node is selected.

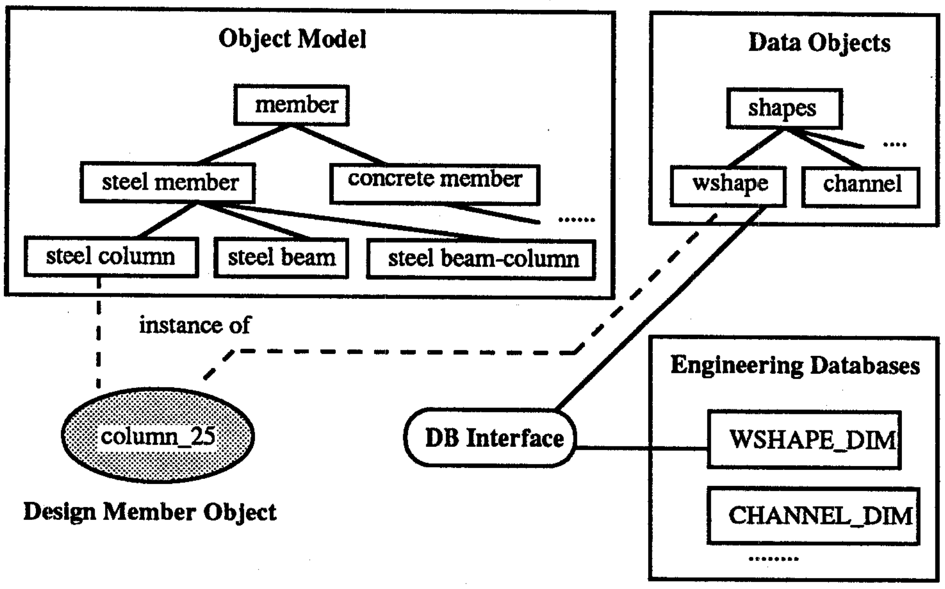


Figure 2 The CAD Object Data Base

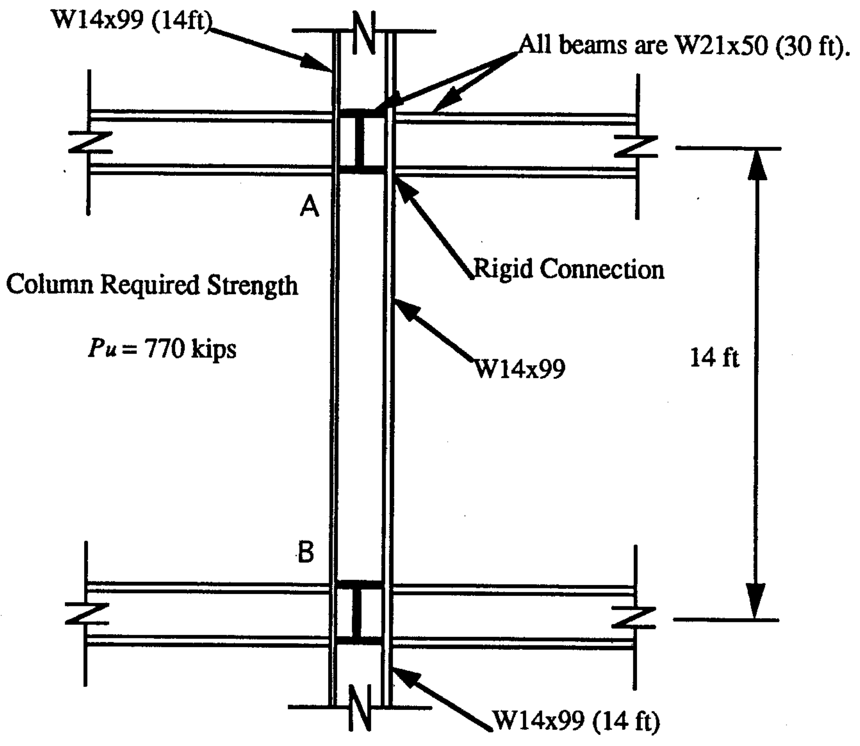


Figure 3 The Column in the Detailed Design Phase

A provision is represented as a Method Object. Each Method Object corresponds to a single data item, and contains a method to compute a value of that data item. There are three types of Method Objects:

- requirements, which check a given design situation and deduce the conclusion of either "satisfied" or "violated,"
- determinants, which determine the value of a data item on which other methods depend, and
- classifications, which classify a design member object into a more specific subclass at each OR node.

Requirements are attached only at the leaf nodes of the AND-OR tree.

In the Object-Logic model, methods are written in terms of Object-Logic sentences expressed as:

A :- $C_1, C_2, \ldots\ldots\ldots, C_n$.

A is a conclusion written in the form:

"Name of Method"(*Term*$_1$, *Term*$_2$, ..., *Term*$_m$)

where *Terms* are either variables, object constants, or functional expressions. C_i ($i = 1, 2, \ldots, n$) is a condition which has the following structure:

Mem::*"Name of Method"* <- *"Message,"*
self <- *"Message,"*
or an arithmetic expression.

"*Name of Method*" is a method attribute value of the object variable "*Mem.*" "*Message*" is a literal which has the same form as the conclusion *A*. The symbol "<-" means that the "*Message*" is to be sent to the "*Method Object*" or "self"; and "self" denotes the Method Object itself. Since both Object-Logic sentences and design provisions are of descriptive nature, provisions can be translated into Object-Logic sentences quite easily. Given a set of Object-Logic sentences, conclusions can be deduced by the inferencing mechanism of resolution and message passing among the Method Objects.

For the example shown in Figure 4, given the design member object "column_25," the system traverses the object hierarchy in a depth-first manner as (using the alphabets as denoted in the figure):

A -> B -> D -> H -> K -> M -> Q -> V -> K -> N -> S -> H -> L -> O -> P.

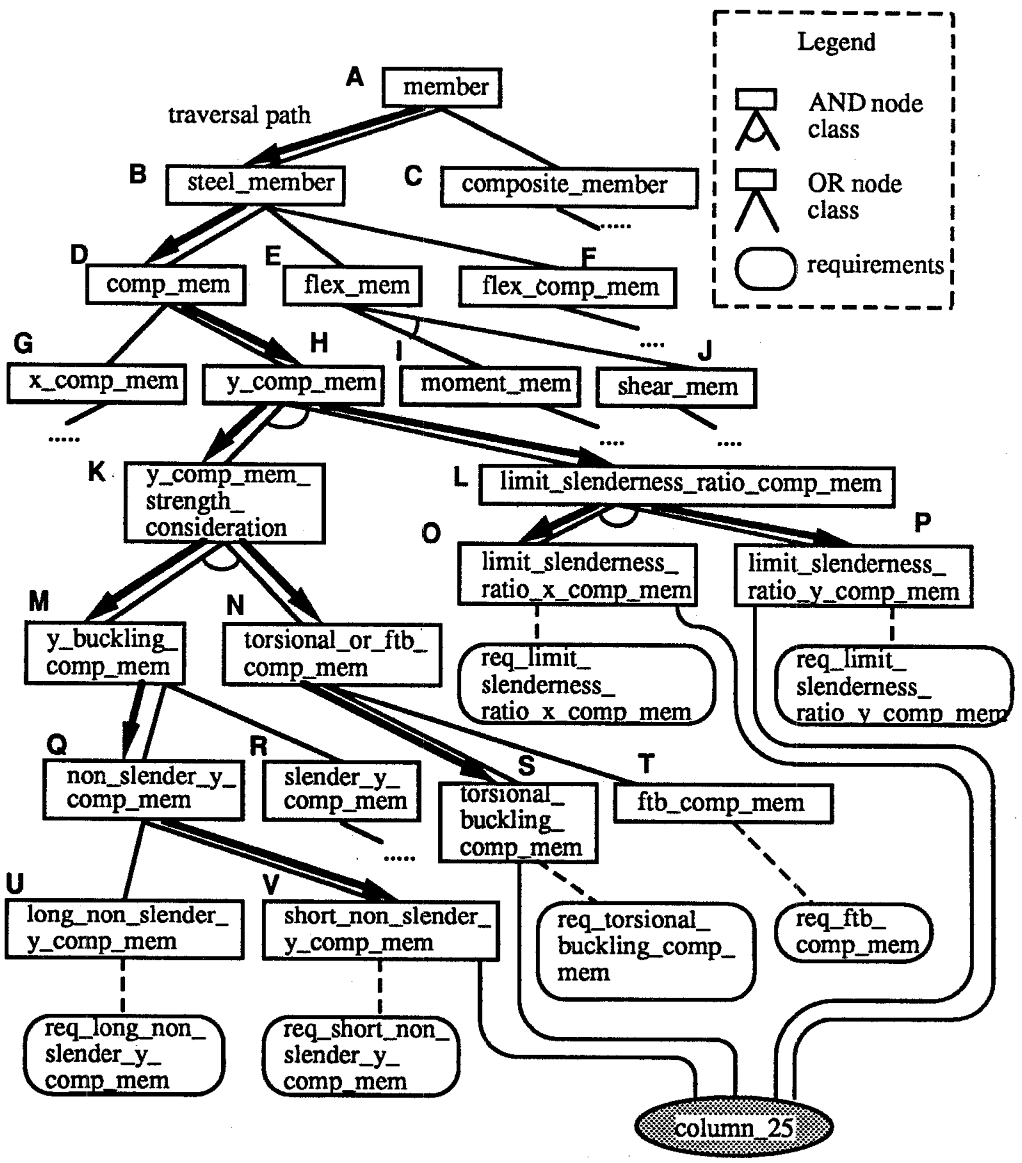

Legend
AND node class
OR node class
requirements
traversal path
A member
B steel_member
C composite_member
D comp_mem
E flex_mem
F flex_comp_mem
G x_comp_mem
H y_comp_mem
I moment_mem
J shear_mem
K y_comp_mem_ strength_ consideration
L limit_slenderness_ratio_comp_mem
M y_buckling_ comp_mem
N torsional_or_ftb_ comp_mem
O limit_slenderness_ ratio_x_comp_mem
P limit_slenderness_ ratio_y_comp_mem
req_limit_ slenderness_ ratio_x_comp_mem
req_limit_ slenderness_ ratio_y_comp_mem
Q non_slender_y_ comp_mem
R slender_y_ comp_mem
S torsional_ buckling_ comp_mem
T ftb_comp_mem
U long_non_slender_ y_comp_mem
V short_non_slender_ y_comp_mem
req_torsional_ buckling_comp_ mem
req_ftb_ comp_mem
req_long_non_ slender_y_ comp_mem
req_short_non_ slender_y_ comp_mem
column_25

Figure 4 Object-Oriented Organization Model with AND-OR Tree

For each leaf node visited, the system stores the requirements linked to the leaf node classes. At the end of the traversal process, the design member object "column_25" is linked to the leaf nodes:

V, S, O, and P.

The requirements to be checked for the design object thus include:

- req_short_non_slender_y_comp_mem (linked to V)
- req_torsional_buckling_comp_mem (linked to S),
- req_limit_slenderness_ratio_x_comp_mem (linked to O), and
- req_limit_slenderness_ratio_y_comp_mem (linked to P),

where the prefix "req_" denotes the requirement for a specific behavior limit state of the member type.

2.3 Conformance Checking Module

Conformance checking refers to the evaluation of a designed member whether it satisfies all the applicable requirements as defined in a design standard. For detailed design, the system identifies all the applicable requirements and automatically executes these requirements based on the given attribute values and external constraints of the member.

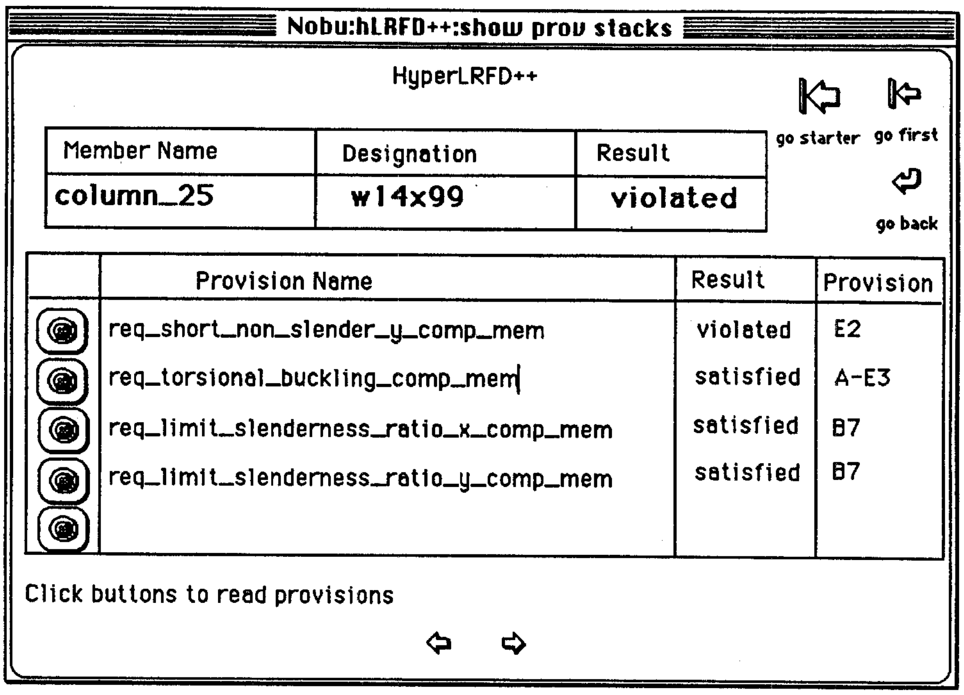
In our prototype system, the user (for example, in a preliminary design process) can focus on checking a single or limited number of specific requirement(s) for a specific object type, stress state, or limit state by selecting a specific class in the Member Class Hierarchy. This procedure reduces the system's execution time and interactions between the user and the system. That is, the user can prune the unnecessary or unimportant requirements that are not of primary concerns in the preliminary design of the member.

In conformance checking, the user first defines a design member object and selects a design strategy, that is either preliminary or detailed design. Then, the system links the design member object to the engineering databases and retrieves the design data of the member. The system traverses the Member Class Hierarchy and records all the applicable requirements. Finally, the system executes the applicable requirements and reports the result to the user. Figure 5 and 6 show, respectively, the conformance checking result and a segment of the design report for the "column_25" shown in Figure 3.

2.4 Component Design Module

Conformance checking is generally easier than producing a design description, because conformance checking has only two possible solutions (i.e., satisfied or violated), while multiple solutions could exist in a design problem. In the Object-Logic model, the design approach is to generate a plausible component design using a set of heuristics and to test the design for conformance checking. If the design violates a requirement or external constraints, the system generates another plausible design and tests it. The design heuristics are procedures applicable for a group of members governed by appropriate requirements and are expressed in Object-Logic sentences.

For example, for selecting a least weight W shape section as a compression member, one heuristic strategy is based on the "column W shapes" table given in the LRFD Manual [AISC 86]. The table consists of a set of designations, effective lengths of the y-axis, and corresponding design compressive strengths based on the limit state of member buckling about the y-axis. Table 1 shows an example of the table of W14 sections with yield stress $F_y = 36$ ksi. For the example of the "column_25," if we assume that the effective length factor of the weak y-axis $K_y = 1.2$ and the



Member Name	Designation	Result
column_25	w14x99	violated

	Provision Name	Result	Provision
	req_short_non_slender_y_comp_mem	violated	E2
	req_torsional_buckling_comp_mem	satisfied	A-E3
	req_limit_slenderness_ratio_x_comp_mem	satisfied	B7
	req_limit_slenderness_ratio_y_comp_mem	satisfied	B7

Click buttons to read provisions

Figure 5 The Conformance Checking Result of "Column_25"

1. Problem Description

Task:	conformance checking
Member Name:	column_25
Designation:	w14x99

Attribute	Value	Unit
elastic_modulus	is 29000.	ksi
shear_modulus	is 11000.	ksi
yield_stress	is 36.	ksi
unbraced_length_x	is 168.	in
unbraced_length_y	is 168.	in
effective_length_factor_x	is 1.6.	
effective_length_factor_y	is 1.2.	

(The rest is omitted.)

2. Design Strategy

Detailed design

3. Result

Requirement	Result
req_short_non_slender_y_comp_mem	violated
req_torsional_buckling_comp_mem	satisfied
req_limit_slenderness_ratio_x_comp_mem	satisfied
req_limit_slenderness_ratio_y_comp_mem	satisfied

Figure 6 Design Report of "Column_25"

Table 1 Design Compressive Strength for W14 Columns (F_y = 36 ksi)

KyLy (ft)	....	W14x90	W14x99	W14x109	....
0	....	810	890	979	....
6	....	795	873	960	....
7	....	789	867	950	....
8	....	783	860	946	....
9	....	775	852	937	....
10	....	767	843	927	....
11	....	758	833	917	....
12	....	749	823	905	....
13	....	738	811	893	....
14	....	728	799	880	....
15	....	716	787	866	....
16	....	704	773	852	....
17	....	691	759	837	....
18	....	678	745	821	....
19	....	664	730	804	....
....	....	....	....	....	....

effective length K_yL_y = 16.8 ft, the W14x109 section can be selected as a candidate since its design compressive strength for an effective length of 17 ft is 837 kips.

2.5 Standards Analysis Module

The Standards Analysis Module is implemented in the form of Object-Logic sentences to check completeness and uniqueness (the lack of redundancy and the lack of contradiction) of a set of rules in the Method Object. This module can be used to check completeness and uniqueness at both the provision and the organizational levels. Checking the properties of the requirement and determinant Method Objects is performed at the provision level. Checking the classification Method Objects allows us to examine the organization of the standard. Completeness of the classification methods ensures that the organization of the standard is collectively exhaustive at every AND node level and uniqueness ensures that the organization is mutually exclusive at every OR node to guarantee that each requirement is uniquely described by a single path in the hierarchy. The testing method is based on the procedure proposed by Jain et al. [Jain 89]. This module parses the Object-Logic sentences, formulates testing sentences, and then tries to prove the validity of the testing sentences using the resolution principle. It also checks whether the relationships among the Method Objects are connected and acyclic. Detailed description of this module is given in Reference [Yabuki 92].

3. The HyperDocument Model

The HyperDocument model is a generic document storage and retrieval model, following the current technology of HyperFile [Clifton 88]. HyperFile is intended to store multimedia documents containing images, graphics, or audio, to support hypertext applications, and to provide a shared repository for multimedia and diverse applications.

The HyperDocument model contains the provisions, background information of standards, external programs, and Method Objects. The documents and programs stored in the HyperDocument model can help engineers to design through easy access to external programs and embedded Method Objects and serve as a large document storage system for the design provisions and background information.

The HyperDocument model consists of a Document Base and a Navigation system. The Document Base contains the documents, each of which consists of its content and HyperTag, as shown in Figure 7. The content of each document is shown and processed by its rendering software such as a word processing software. The HyperTag is a tag that contains information about the document such as title, author, file name, rendering software, and pointers to other documents. The Navigation system facilitates document retrieval, and provides three basic navigation mechanisms: pointers, queries, and browsers.

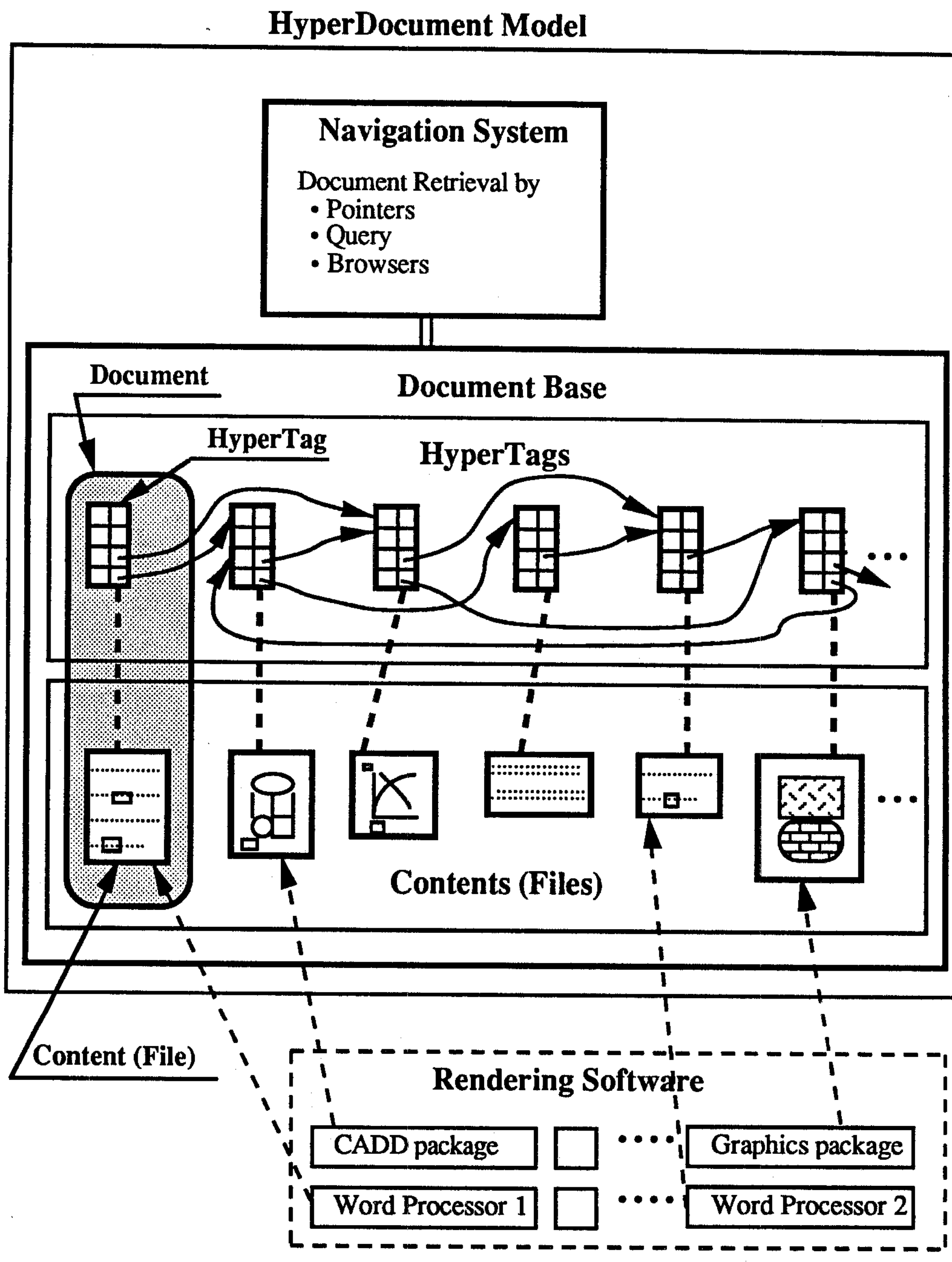


Figure 7 The Overview of the HyperDocument Model

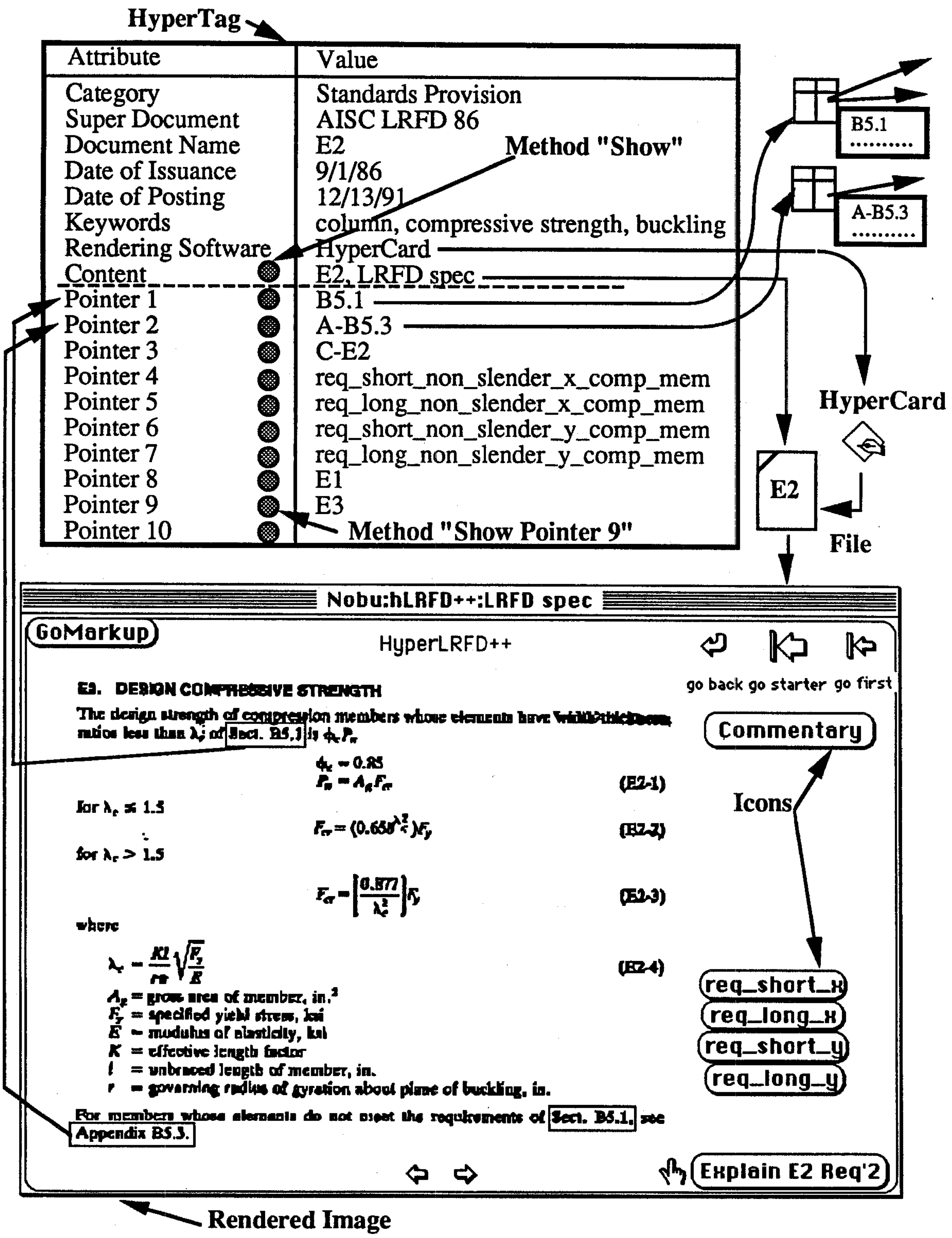
The Document Base of the HyperDocument system for design standards consists of the following four document clusters:

- Method Objects, which are Object-Logic programs representing provisions,
- Provision Document Base, which contains provision documents,
- Background Base, which contains a large amount of background information related to design standards such as explanations and commentaries to the provisions and research papers, and
- External Programs, which process charts or complex equations that appear in the standard and background information to derive necessary data items.

When creating or revising a standard, a large amount of information in various forms may be collected. Such information as well as explanations about the provisions can be stored in the Background Base. The documents in the Background Base can then be used by code developers for future revisions and by engineers for obtaining explanations and background information about the provisions. During the design process, many data items require complex calculations involving graph or chart processing. External programs perform such tasks and return the computed results to the design program. Such programs, such as spreadsheet programs, can be accessed through the HyperDocument system.

In the example of the "column_25" shown in Figure 5, the requirement "req_short_non_slender_ y_comp_mem" is violated. By clicking the button next to the requirement, the corresponding provision "E2" is popped up as shown in Figure 8. Note that HyperTag is behind the rendered image of the document file. This provision is referencing to provisions "Sect B5.1" and "Appendix B5.3." By the pointers from the HyperTag of the provision "E2" to these referenced documents, we can retrieve the documents corresponding to these provisions. In addition, the provision "E2" has pointers to other documents such as commentary, explanation of the provision, and the corresponding Method Objects.

In the example shown in Figure 3, the effective length factor K_x of "column_25" is unknown. To compute this value, the user can request the system to show the provision explaining the effective length factor. By the pointer from the Method Object "k_factor_x" to the provision document "C2," the system shows us the provision as in Figure 9. The user can further read commentary "C-C2" by clicking the "Commentary" button on the provision card. As the user browses through the commentary cards, the alignment chart for determining the effective length factor K of unbraced frame appears on the screen as shown in Figure 10. As noted in the lower corner of Figure 10, there is as external program that can be used to calculate the K-factor. The user can click the button "K-factor" to open the spreadsheet program for determining K as shown in Figure 11. As shown in Figure 11, based on the input data(designations, lengths, and considering axes of the columns and beams) provided, the spreadsheet program automatically computes the restraint factors G_A and G_B for the end joints of the column and the effective length factor K. The K_x value is then sent to the Method Object "k_factor_x" after the spreadsheet program is closed. The effective length factor K_y can be calculated in a similar manner. (In this example, the K_x and K_y values are determined to be 1.6 and 1.2 respectively.)



Attribute	Value
Category	Standards Provision
Super Document	AISC LRFD 86
Document Name	E2
Date of Issuance	9/1/86
Date of Posting	12/13/91
Keywords	column, compressive strength, buckling
Rendering Software	HyperCard
Content ●	E2, LRFD spec
Pointer 1 ●	B5.1
Pointer 2 ●	A-B5.3
Pointer 3 ●	C-E2
Pointer 4 ●	req_short_non_slender_x_comp_mem
Pointer 5 ●	req_long_non_slender_x_comp_mem
Pointer 6 ●	req_short_non_slender_y_comp_mem
Pointer 7 ●	req_long_non_slender_y_comp_mem
Pointer 8 ●	E1
Pointer 9 ●	E3
Pointer 10 ●	

Nobu:hLRFD++:LRFD spec

HyperLRFD++

E2. DESIGN COMPRESSIVE STRENGTH

The design strength of compression members whose elements have width-thickness ratios less than λ_r of Sect. B5.1 is $\phi_c P_n$

$$\phi_c = 0.85$$

$$P_n = A_g F_{cr} \quad \text{(E2-1)}$$

for $\lambda_c \leq 1.5$

$$F_{cr} = (0.658^{\lambda_c^2}) F_y \quad \text{(E2-2)}$$

for $\lambda_c > 1.5$

$$F_{cr} = \left[\frac{0.877}{\lambda_c^2}\right] F_y \quad \text{(E2-3)}$$

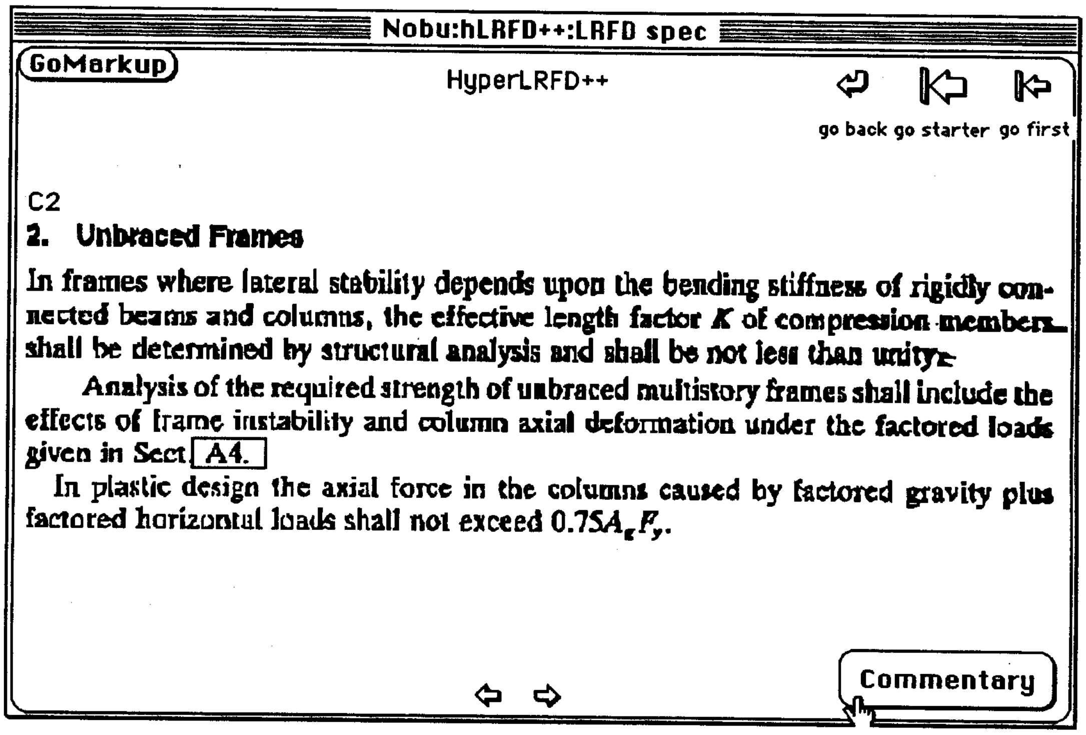
where

$$\lambda_c = \frac{Kl}{r\pi}\sqrt{\frac{F_y}{E}} \quad \text{(E2-4)}$$

A_g = gross area of member, in.^{2}
F_y = specified yield stress, ksi
E = modulus of elasticity, ksi
K = effective length factor
l = unbraced length of member, in.
r = governing radius of gyration about plane of buckling, in.

For members whose elements do not meet the requirements of Sect. B5.1, see Appendix B5.3.

Figure 8 A Sample Document in the Document Base



C2

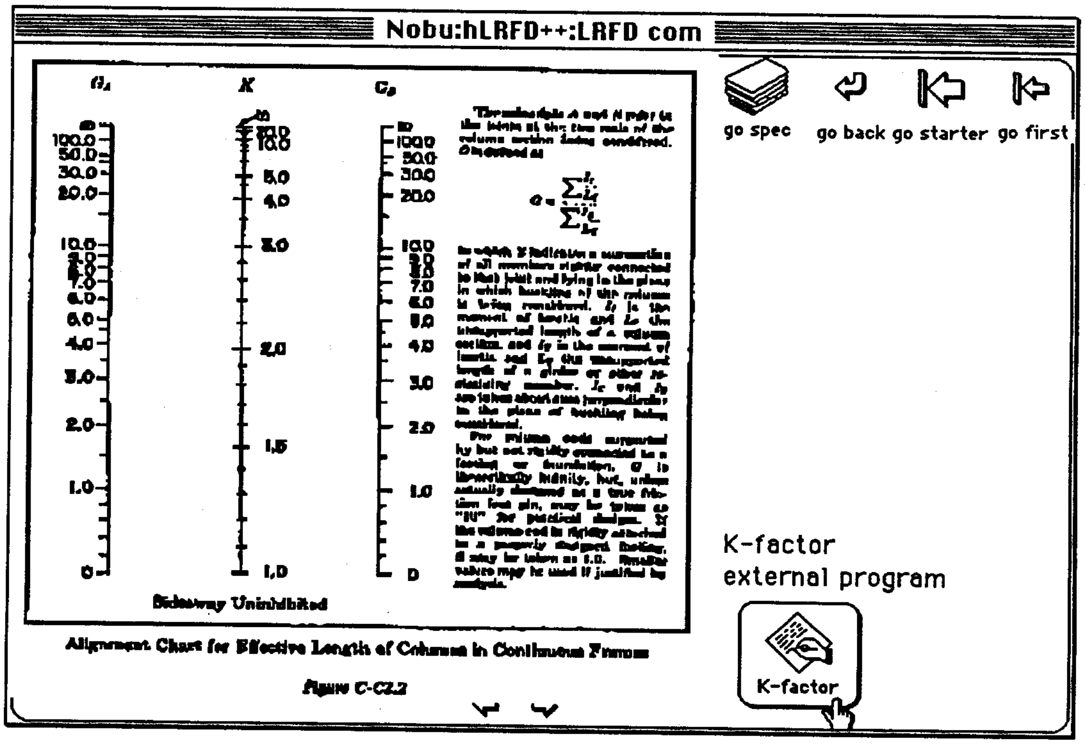
2. Unbraced Frames

In frames where lateral stability depends upon the bending stiffness of rigidly connected beams and columns, the effective length factor K of compression members shall be determined by structural analysis and shall be not less than unity.

Analysis of the required strength of unbraced multistory frames shall include the effects of frame instability and column axial deformation under the factored loads given in Sect. A4.

In plastic design the axial force in the columns caused by factored gravity plus factored horizontal loads shall not exceed $0.75A_gF_y$.

Figure 9 Provision Document "C2"



The subscripts A and B refer to the joints at the two ends of the column section being considered. G is defined as

$$G = \frac{\sum \frac{I_c}{L_c}}{\sum \frac{I_g}{L_g}}$$

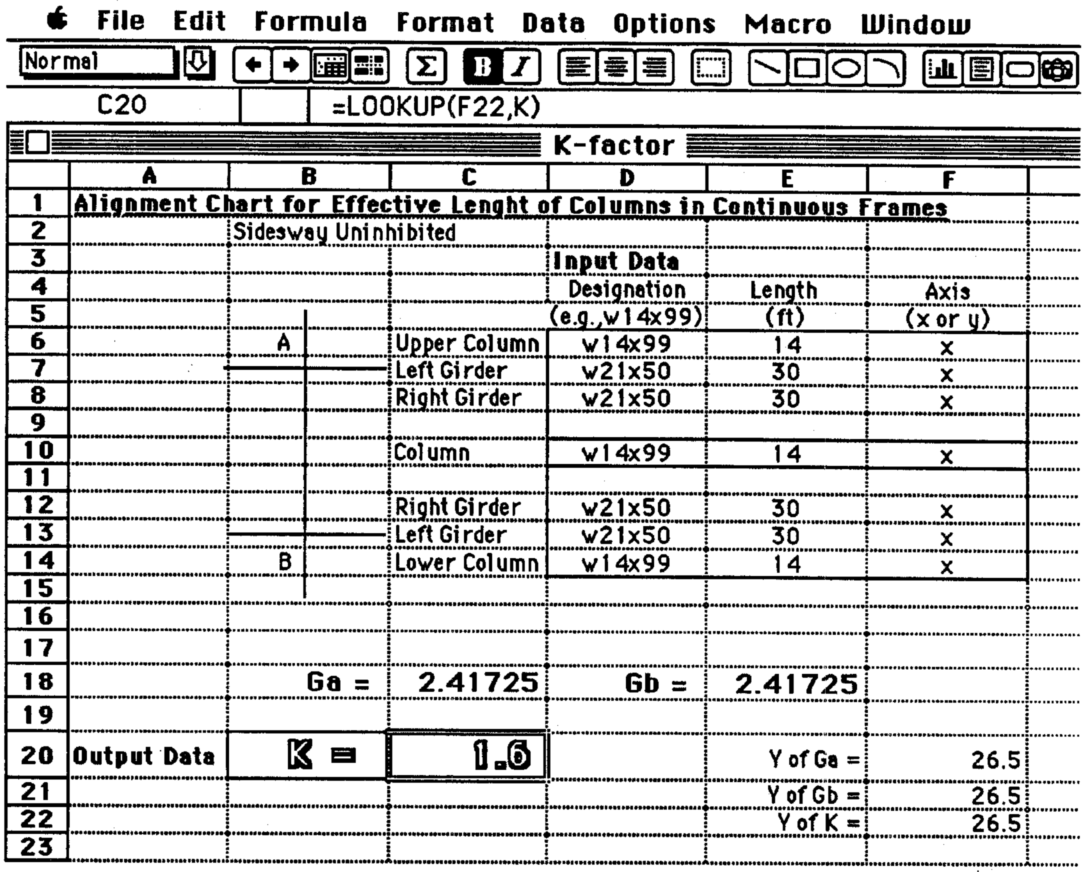
in which Σ indicates a summation of all members rigidly connected to that joint and lying in the plane in which buckling of the column is being considered. I_c is the moment of inertia and L_c the unsupported length of a column section, and I_g is the moment of inertia and L_g the unsupported length of a girder or other restraining member. I_c and I_g are taken about axes perpendicular to the plane of buckling being considered.

For column ends supported by but not rigidly connected to a footing or foundation, G is theoretically infinity, but, unless actually designed as a true friction free pin, may be taken as "10" for practical design. If the column end is rigidly attached to a properly designed footing, G may be taken as 1.0. Smaller values may be used if justified by analysis.

Alignment Chart for Effective Length of Columns in Continuous Frames

Figure C-C2.2

Figure 10 The Document Showing the Alignment Chart with a Button



C20 =LOOKUP(F22,K)

K-factor

	A	B	C	D	E	F
1	Alignment Chart for Effective Lenght of Columns in Continuous Frames					
2		Sidesway Uninhibited				
3				Input Data		
4				Designation	Length	Axis
5				(e.g.,w14x99)	(ft)	(x or y)
6		A	Upper Column	w14x99	14	x
7			Left Girder	w21x50	30	x
8			Right Girder	w21x50	30	x
9						
10			Column	w14x99	14	x
11						
12			Right Girder	w21x50	30	x
13			Left Girder	w21x50	30	x
14		B	Lower Column	w14x99	14	x
15						
16						
17						
18		Ga =	2.41725	Gb =	2.41725	
19						
20	Output Data	K =	1.6		Y of Ga =	26.5
21					Y of Gb =	26.5
22					Y of K =	26.5
23						

Figure 11 The Screen Image of the K-factor Program of Excel

4. Summary and Discussion

In this paper, we proposed an Object-Logic model for design standards processing. This model combines the object-oriented and logic programming paradigms to represent the organization and provisions of a design standard. In addition, we described a HyperDocument model, which serves as a heterogeneous document storage and manipulation system useful for code developers and design engineers. The Object-Logic and the HyperDocument models were integrated into a unified Hyper-Object-Logic model. In this model, the user can perform both conformance checking and component design as well as checking completeness and consistency of design standards. This model can be integrated with other design applications.

To demonstrate the feasibility and practicality of the Hyper-Object-Logic model, a prototype system, HyperLRFD++, has been implemented for the AISC LRFD specification [AISC 86]. HyperLRFD++ has been implemented on an Apple Macintosh IIsi computer. The following software packages are employed in the implementation:

- Prolog++ [Quintus 90], an extension of Prolog with a full object-oriented programming environment, is used to implement the Conformance Checking Module, Component Design Module, Standards Analysis Module, Standards Base, Object Data Model, and Member Object.
- HyperCard [Apple 90], a Hypertext software for Macintosh, is used to implement HyperTags and documents in the HyperDocument system and the user interface.

- Oracle [Oracle 89], a relational database system, is used to implement the dimensions and properties of wide-flange W shape sections.
- MacDBI [Quintus 91], a system interface between Prolog and the Oracle database system for Macintosh, is used to integrate the Object Data Model and the engineering databases within the CAD Object Data Module.
- Excel [Microsoft 91], a spreadsheet software, is used to implement the external program that processes graphs and charts such as the alignment chart for determining the effective length factor.

HyperLRFD++ incorporates the sections of the AISC LRFD specification that are related to compression members, flexural members, and members subject to bending and compression (i.e., beam-columns). For the demonstration purpose, only wide-flange W sections are included in the database. HyperLRFD++ is capable of component design and conformance checking of W shape section members. Furthermore, the prototype system can be used to analyze the design standard. This demonstration system has been used to study the viability and effectiveness of the Hyper-Object-Logic model for design standards processing.

Acknowledgment

Information in Figures were optically scanned and reproduced from the AISC specification [AISC 86] with permission of the publisher, the American Institute of Steel Construction, Inc. We would like to acknowledge the sponsorship for the graduate study of Nobuyoshi Yabuki at Stanford University by the Electric Power Development Co., Ltd. and the Japan-U.S. Educational Commission (Fulbright Scholarship). This research is also partially supported by the National Science Foundation, Grant No. IRI-9116646.

References

[AISC 86] American Institute of Steel Construction, Inc., "Manual of Steel Construction — Load & Resistance Factor Design," First ed,. 1986.

[Apple 90] Apple Computer Inc., "HyperCard Basics," HyperCard Manual, 1990.

[Clifton 88] Clifton, C., Garcia-Molina, H., and Hagmann, R., "The Design of a Document Database," Technical Report No. CS-TR-177-88, Department of Computer Science, Princeton University, 1988.

[Clifton 90] Clifton, C. and Garcia-Molina, H., "Distributed Processing of Filtering Queries in HyperFile," Technical Report No. CS-TR-295-90, Department of Computer Science, Princeton University, 1990.

[Garrett 86] Garrett, Jr., J. H. and Fenves, S. J., "A Knowledge-Based Structural Component Design," Report No. R-86-157, Department of Civil Engineering, Carnegie-Mellon University, 1986.

[Jain 89] Jain, D., Law, K. H., and Krawinkler, H., "On Processing Standards With Predicate Calculus," *Proceedings of the Sixth Conference on Computing in Civil Engineering*, Atlanta, GA, pp.259–266, 1989.

[Microsoft 91] Microsoft Corporation, "Microsoft Excel 3.0 Reference," 1991.

[Oracle 89] Oracle Corporation, "ORACLE for Macintosh Reference," 1989.

[Quintus 90] Quintus Computer Systems, Inc., "Quintus Prolog++ Reference Manual," 1990.

[Quintus 91] Quintus Computer Systems, Inc., "MacDBI for Oracle Reference Manual," 1990.

[Yabuki 92] Yabuki, N., "An Integrated Framework for Design Standards Processing," Ph.D. Thesis (under preparation), Department of Civil Engineering, Stanford University, 1992.