# CONSTRUCTING AN INTEGRATED CLIENT/SERVER FRAMEWORK FOR OPERATIVE CHECKING OF BUILDING CODE

**Svetla Vassileva**
University of Architecture Civil Engineering and Geodesy,
Department of Computer Aided Engineering, Sofia, Bulgaria,
E-mail: Svetla_fce@uacg.acad.bg

*ABSTRACT: The paper presents results from a number of investigations into the problem of the using an integrated client/server framework for an automated code-checking system. The changing nature and the complexity of building codes leads to delays in the design and the construction processes. The designer must assess which codes are applicable to a given project. Inspectors must go through a similar process and there can be inconsistencies in interpretation of a given section of the code between different inspectors. After that, he must check the codes for potential ambiguities in the code provisions. The process of design checking and approval can prolong the construction and delay the operation of a facility. Automating this process can alleviate the inconsistencies and delays with manual checking. Most previous studies on the process of checking of building codes have focused on the processing of design codes for conformance checking. The present article proposes to add additional criteria to a building model. On that basis are summarized representations of code provisions. The structure and attributes of a product model and building code model needed to provide design information are examined by a code-checking program. This program can read the design data and reorganize the information in a form that can be analyzed and compared to the model of the building code. The building code model is described as a mapping of building code provisions in an object-oriented framework. A program for analysis of a building design is developed to automate the process of checking a building design for compliance to a building code document. AutoCAD is used as a design environment. The building model is based on IFC Release 1.5 and on an additional layer of building component objects. This layer is created with semantics corresponding to the IFC specifications. During the process of design the designer can send the building model to the code-checking program. A program in Auto Lisp extracts the IFC information from the AutoCAD database and converts the information into an IFC EXPRESS file. The building code model is based on the same structure as the IFC project model hierarchy. The code-checking program reads in a stream of IFC data to populate its database of building components. The program reads in a stream from a building code file, which is a mapping from the text of the provisions of a building code document to an EXPRESS file. The code-checking program is on the server of a client/server framework. This program reads in a building code EXPRESS file and populates a data structure containing instances of the building code provisions. Finally, the system determinates if a set of provision is relevant to the specific building component associated with a specific space.*

*KEYWORDS: Client/Server Framework, Automated Code-Checking System, Building design, Database, Computer - Aided Design.*



## 1. INTRODUCTION

The changing nature and the complexity of building codes leads to delays in both the design process and the construction process. Construction documents submitted to a building department are checked manually against a continuously changing and increasingly complex set of building codes. The designer must assess which codes are applicable to a given project and review them for potential ambiguity in the code provisions. After that, an inspector must go through a similar process. Inconsistencies can arise since the interpretation of a given section of the code may be different from inspector to inspector. The design checking and approval process can be a critical activity that prolongs the construction and delays the operation of a facility. Automating this process has the potential to alleviate both the delays and the inconsistencies associated with manual checking by giving the designer a consistent framework in which to apply and check codes.

Several researches have developed frameworks for the representation and processing of design standards (de Waard 1992, Yabuki 1992). Kiliccote (1996) examines different aspects of the building code. A survey of developments for computer representation of design codes was given by Fenves (1995). The authors focuses on the problem of developing a framework for architectural building code issues. In this study the representation and processing of design standards are based not only on architectural, but as well on constructive building code issues

Web-based technologies and advances in the Internet will have a significant role in making operative checking of building codes reality. On-line checking of building design via the World Wide Web can be organized in a client-server environment. The development of a standard product model including the Standard for Industry Foundation Classes (IFC) ("Industry" 1998) and Exchange of Product Data (STEP) ("Industrial" 1994) will further facilitate design data exchange. Based on the first of these works, the client develops a plan using an IFC-compliant CAD package. In this package, enhancements have been made to AutoCAD to output an IFC EXPRESS file from a building design.

In the process of design, the user can send this project to a code-checking program that resides on a remote server. This code-checking program examines the IFC design data and summarizes the results in a generated web page, which contains a graphical representation of the building model along with information with hyperlink to specific comments. These comments have hyperlinks to the actual building code document provisions.

Examination of the structure and attributes of a product model and a building code model was provided to analyze sufficient design information by a code-checking program. The code-checking program must be able to read the design data and reorganize the information in a form that can be analyzed against the model of the building code. The building code model is described in this paper as a mapping of building code provisions to methods that are embedded in an object-oriented framework appropriate to analyze a design process for code compliance. The building code can be structured into three classes:

- Provisions that examine the criteria of a system of building components.
- Provisions that examine the criteria of individual building components.
- Provisions that determine the relevancy of other provisions.

This paper is concerned the problems connected with operative checking of building code based on integration of a building product model and the object-oriented building product model.

## 2. REPRESENTATION OF THE BUILDING PRODUCT MODEL

The program that does the analysis of a building design must understand the design data for automation of the checking of a building design in compliance with a building code document. An inspector in a building department must coordinate related drawings, which are two-dimensional representations of three-dimensional information on paper. The related drawings usually are plans with elevations in order to develop a three-dimensional image to check a design against a building code. At three-dimensional model has several advantages over viewing and interpreting a two-dimensional representation:

- Behavior and function of the building model components can be more accurately modeled in a three-dimensional representation.
- Elimination of the need for developing a three-dimensional building model from two-dimensional views.
- Representation of building components and their geometrical relationships to other components is explicit.

Current CAD systems allow the designer to develop three-dimensional building models, but when designers assemble a design or construction document package, they generate two-dimensional representations for the review process. The projection of the three-dimensional representation to a set of two-dimensional representations loses design data. The simplest is the loss of data of one of the three dimensions. Using three-dimensional building models directly for analysis purposes alleviates the need for the process of projection and subsequent regeneration from two-dimensional back to a three-dimensional model.

The International Alliance of Interoperability is trying to develop standards for a three-dimensional project model that enables interoperability between applications by different software vendors (“Industry” 1998). These standards include defining a set of objects called Industry Foundation Classes (IFC) and according to IFC Release 1.0 there are two standard formats for sharing project data: s standard EXPRESS file format and software interfaces. The definitions of some entities classes were changed in the next IFC Release, 1.5. The building product model in this study is based on IFC Release 1.5. AutoCAD Release 14 is employed as a design environment. The implemented IFC class hierarchy is shown in figure 1.

Since AutoCAD doesn’t currently support the IFC building model, a simple AutoCAD to IFC translation module has been developed. This module generates the building model class attributes and relationships required by the code-checking program to perform the building code analysis. AutoCAD is used to create an additional layer of building component objects with semantics corresponding to the IFC specification. An example of IFC relationships of building components is shown in figure 2. During the design process, the designer can send the building model to the code-checking program that resides on a remote server. A program in AutoLisp extracts the IFC information from AutoCAD database and converts the information into an IFC EXPRESS file. The code-checking program analyzes the building model and summarizes the results in a generated web page, which contains a graphical representation of the building model.

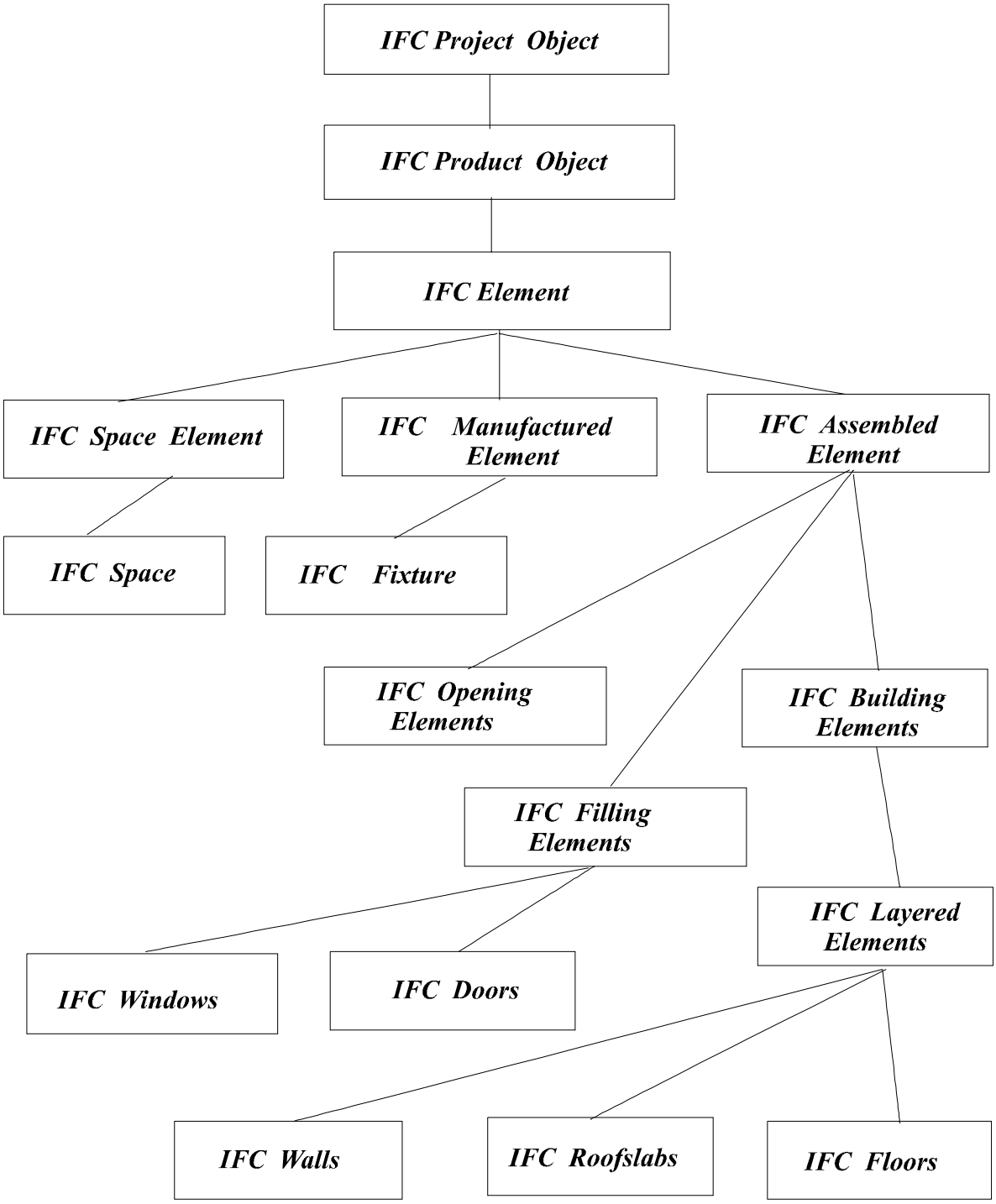


*Fig.1. IFC Class Hierarchy Implemented*

For the code-checking program to correctly analyze the building model, it is necessary to make an assumption about the use of the IFC Space that describes attributes of a space. The designer needs to make explicit whether a space needs to be accessible, as it's shown in section 4.

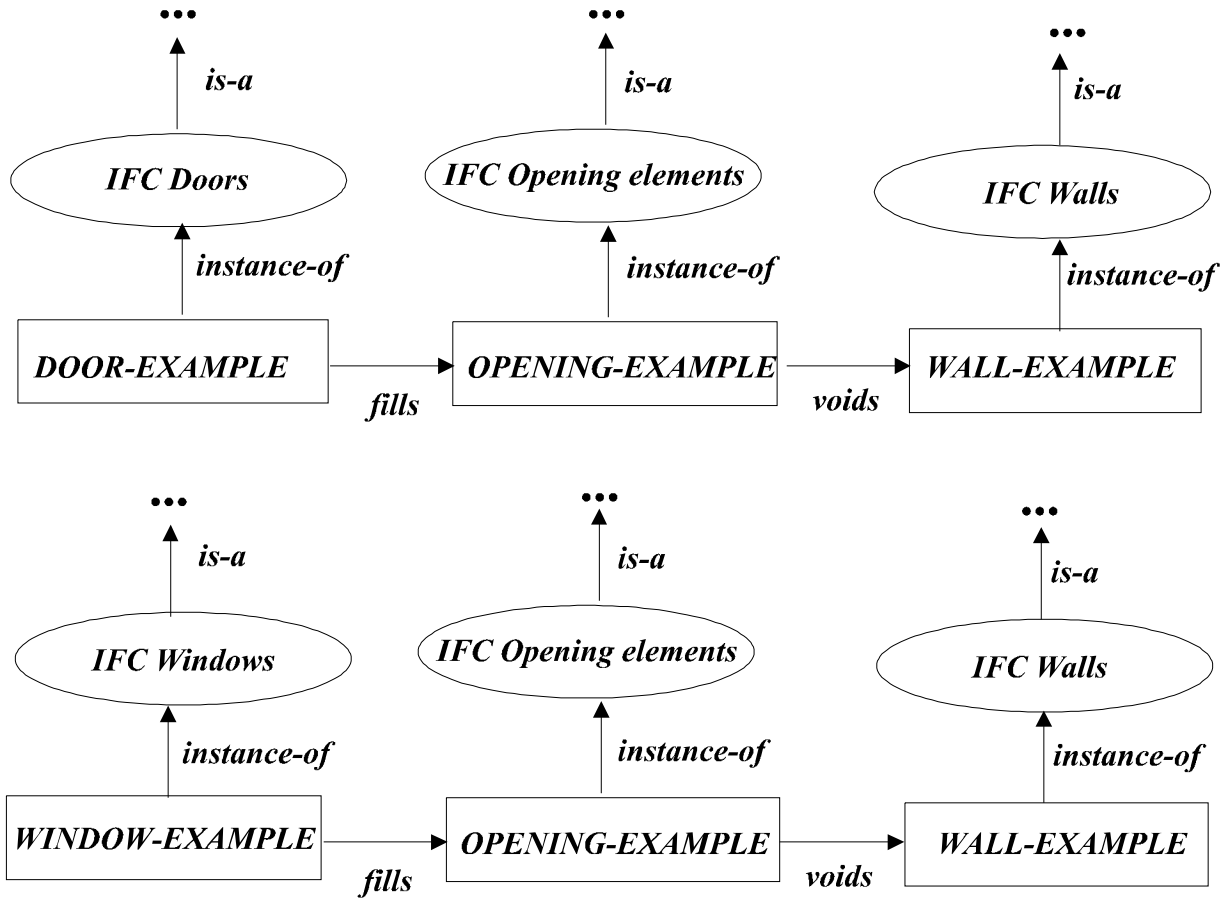


*Fig.2. Example of Relationships between Doors and Walls Classes and Windows and Walls Classes*

## 3. CLIENT/SERVER FRAMEWORK FOR CHECKING OF BUILDING CODE

The code-checking program resides on the server side of the system. The client and the server sides of the system are written in Java but the two sides have system independence. The client sends an IFC EXPRESS file across the network to the code-checking program. The code-checking program continually listens to a predetermined socket on the server for the appropriate start sequence and the subsequent stream of IFC EXPRESS data. A program in AutoLisp spawns the client process after the generation of the IFC EXPRESS file. The mechanics of the prototype code-checking program are described in figure 3. The client program can also be started independently of the AutoLisp program and will execute as long as there is generated IFC EXPRESS file that is ready to be sent to the code-checking program. The user (client) opens the IFC EXPRESS file and the corresponding socket on the server. After that, he sends the appropriate start sequence and then sends the stream of the IFC EXPRESS file to the code-checking program. The results are posted to a web page. The code-checking program notifies the client that the analysis is complete.

The code checking program stores the IFC data as objects in a data structure that uses objects based on the hierarchical structure shown in figure 1. The Java IFC classes are constructed to

directly map the attributes and relationships as defined in the IFC EXPRESS schema. The IFC objects in the data structure have slots to accommodate additional information that is generated by the code-checking program. This information may be additional graphical information and comments associated with either code violations and code compliance.

The code-checking program on the server reads the IFC data sequentially. An object that makes reference to a previously instantiated object will point to the previous object in the appropriate attribute field. Certain attributes of a building component are described by a subsequent building component instantiated on another line in the EXPRESS file. The code-checking program is ready to analyze the building model when all the IFC data has been put into a data structure containing the building components.

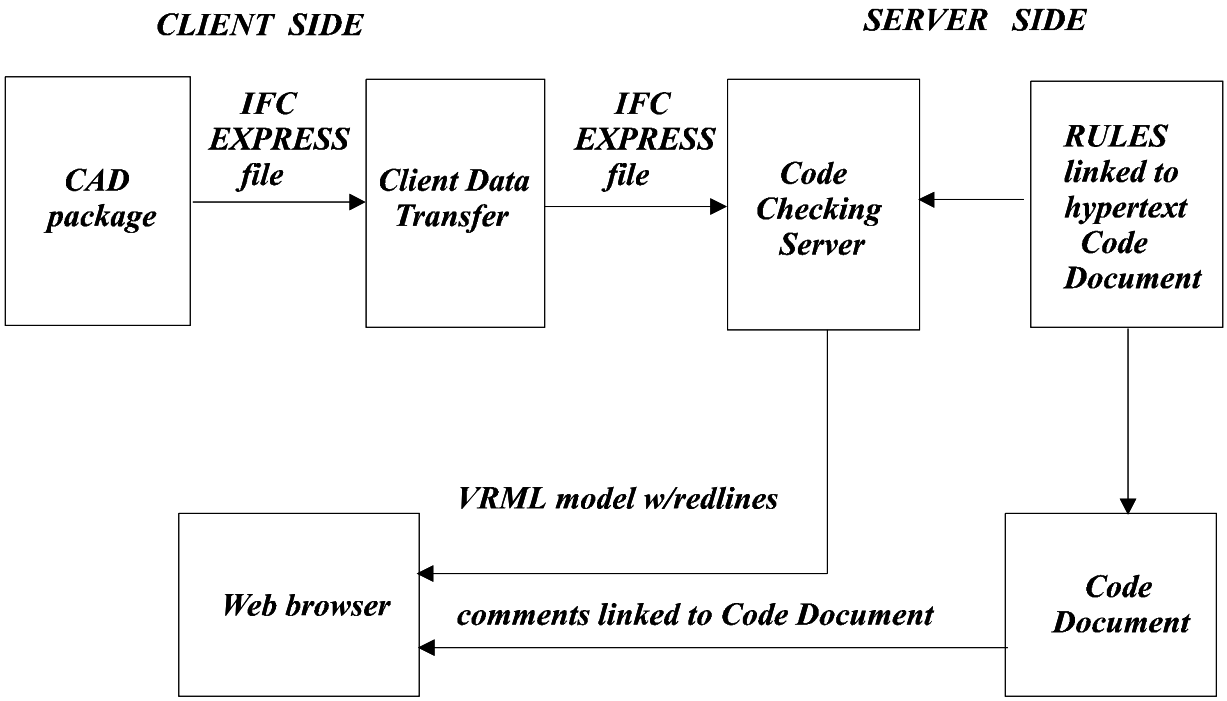


*Fig.3. Mechanics of Prototype Code-Checking Program*

## 4. REPRESENTATION OF THE BUILDING CODE MODEL

The building code model uses the same structure as the IFC project model hierarchy. All provisions concerning certain elements are encoded in a corresponding class;, for example window accessibility would be instances of a window accessibility class. Every individual component can be checked against all applicable instances of the provisions for that class of building component. All components are grouped by similar functional units and similar code issues can be grouped. For example, windows and doors have egress-related provisions. By structuring the encoded provisions in this manner, one can loosely categorize building component provisions by design intent, since the behavior of similar building components is similar.

The code-checking program reads in a stream of IFC data to populate its database of building components. Similarly, the program reads in a stream from a building code file and this file is a mapping from the text of provisions in a building code document to an EXPRESS file that has instances of the encoded provisions. The code-checking program reads in the building code EXPRESS file and populates a data structure containing instances of the building code provisions.

For checking a building design against a building code, "relevance" is applied in the following ways:

- Determining which provisions are applicable to a given building component or system of building components.
- Determining which building component or system of building components are applicable to a given provision.
- Resolving exceptions within a provision.

There are several levels of relevance that need to be addressed. To determine the relevance of provisions for a specific building component, a top down approach is taken and the hierarchy follows the same class hierarchy as the IFC model:

- The system decides if a set of provisions is relevant to the project under the project model.
- The system examines specific buildings, specific floors, and then specific spaces.
- The system determines if a set of provisions is relevant to the specific building component associated with a specific space.

One can determine whether provisions are relevant to a building or project given the same information that building inspectors receive in construction documents. The relevance must determine whether a set of provisions is applicable to a given space in the building product model. Only the relevance issues related to accessibility on the space level are examined in the current implementation of the relevance module. The user is required to label a space from IFC Space as accessible and the code-checking program examines this information to note which spaces are accessible. Every building component can have several states. The code-checking program initially labels a space as either REQUIRED or NOT-REQURED according to how the user labels the space. If the space is REQUIRED the code-checking program determines which building components are associated with a specific space.

The code-checking program analyze the building components in each space that has been labeled REQUIRED and after that the code checker must again determine whether an individual building component from the REQUIRED space is relevant for the overall code compliance of the building design. This is the issue of cardinality, deciding whether a building component needs to be accessible in relation to the other building components in the examined space. Every individual building component must be examined in relation to other building components of the same class in a given space, because higher level provisions often determine the number of similar building components that need to comply to a specific building code issue. The code-checking program must first analyze the building components in a space and then group and reanalyze the subset of similar components to determine whether they finally need to comply with the accessibility code provisions. If a building component doesn't meet accessibility code

compliance, it may not mean that the project is in violation of the accessibility code. The building code may only require that there exist a similar building component in the same space that complies.

## 5. ENCODED BUILDING COMPONENT PROVISIONS

The code-checking program analyzes the building model twice: first it analyze all the building components in the REQUIRED spaces, and than it determines whether a component needs to comply with the building code provisions relative to the other components in the space. The encoding of building component provisions can be mapped to methods in the building code model. Encoding these provisions alone isn't sufficient for the building code framework, because the code-checking program must apply relevance test to the specific building components to see whether a provision or set of provisions is applicable.

The code-checking program generates sets of building components associated with particular spaces that need to be checked for accessibility. Each building component is associated with a set of one or more spaces. The code-checking program establishes these relationships and examines all the building components, including components that are associated with a space that aren't required to be accessible. After that, each component is checked against the relevant provisions. After all related components are checked, the code checker determines whether the set of components is in compliance. The building component initially has a state indicating that the requirement for accessibility is unknown and it's state is set to PASSING unless it fails one of the encoded provisions.

The code-checking program examines each of encoded provisions in the building code instances until it matches the attributes connected with:

- the class of the building component with the attribute "IFC-Class" in the building code component;
- the attribute "level" with the string "ELEMENT".

The code-checking program is examining building components on the element level as opposed to the space level in this inspection of the building model. When the code-checking program finds a match, it tests the encoded provision (most of them are geometric test) against the building component. The building component must be checked against other building components within the associated accessible space for those that are associated with the issue of clearance.

If the building component complies with the encoded provision, the code component returns a PASSING status, and if not, it returns a FAILED status. When the module receives this return status, it continues to traverse and search for relevant encoded provisions. The code-checking program continues to check a component even if it has failed a previous test and it can return as much information on a building component as possible.

After examining all of the building components, the code-checking program must resolve whether an individual building component within an accessible space is required to comply with the accessibility code. After that, the code-checking program makes a second pass through the database of building components looking for matches with code component attribute "IFC-Class"

and the type of building component. It also looks for matches between the code component attribute “level” and the string “SPACE”. The code-checking program concurrently examines a set of identical building components associated with the same space.

When all building code issues have been attached to the building components, the code-checking program has finished the analysis of the building model. After that, the system generates the information in the form of a web page.

## 6. GENERATION OF WEB PAGE

When all building code issues have been attached to the building components, the code-checking program examines the building component database one final time and extracts the necessary information to generate a web page.

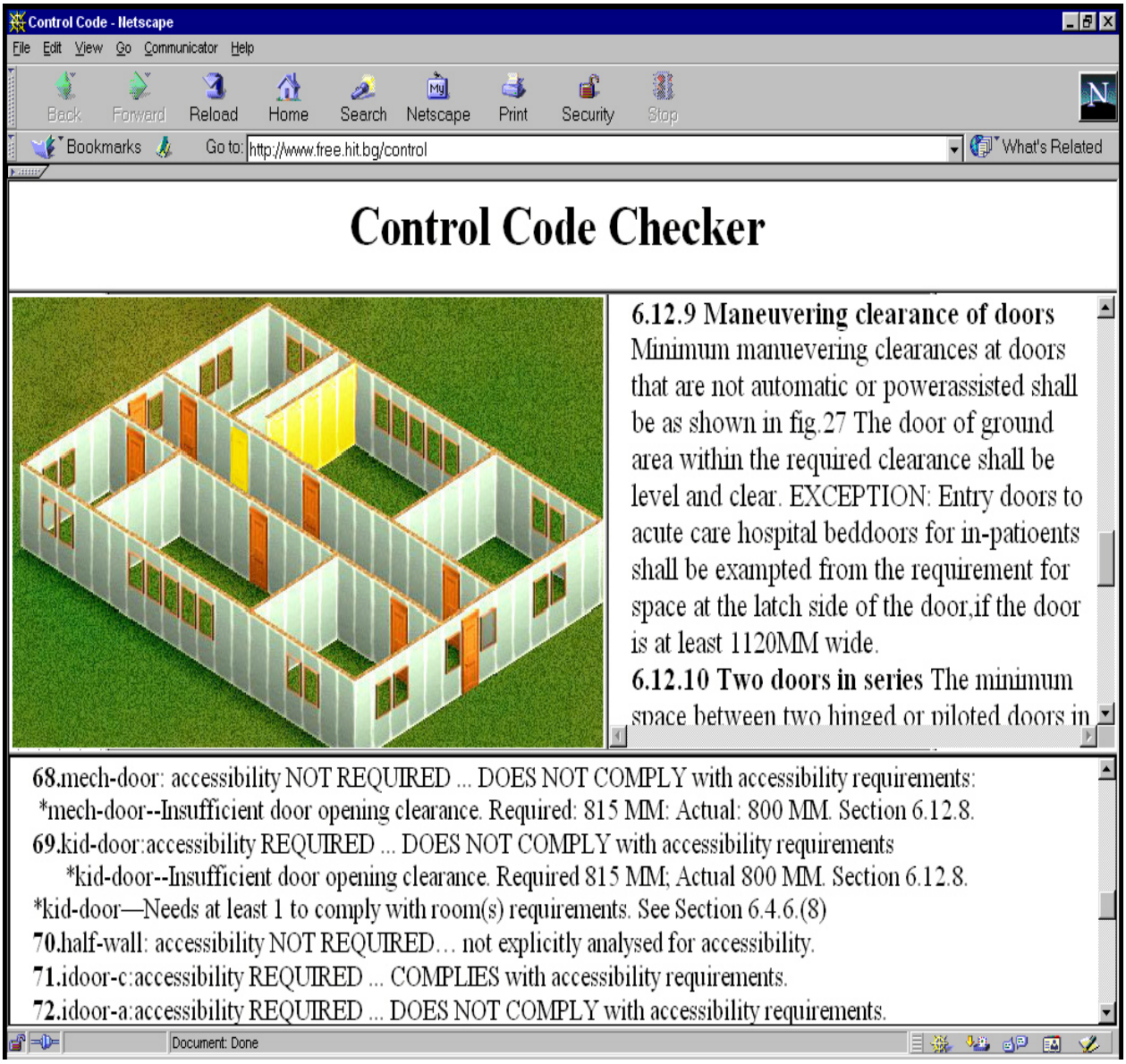


*Fig. 4. Web Page Generation for Noncompliant Design Example*

In figure 4 is shown web page generated for noncompliant building design. This web page consists of three frames:

- VRML frame;
- Comments frame;
- Building code document frame.

In figure 5 is shown web page generated for revised and compliant design.

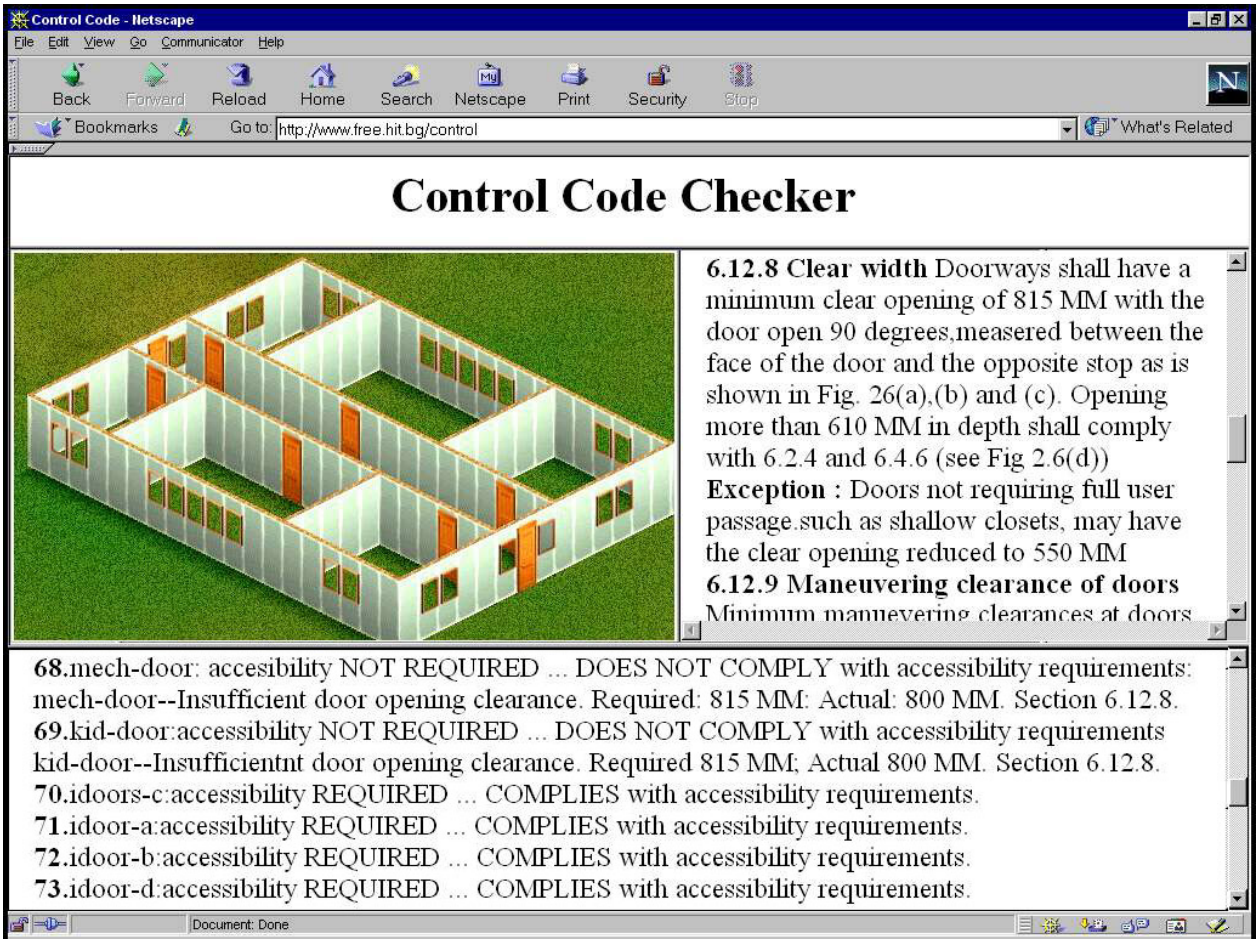


*Fig. 5. Web Page Generation for Revised and Compliant Design Example*

Color-coding a building component is useful for the designer to realize if the specific component isn't accessible and is NOT-REQURED. When the compliance of the building component is FAILED, it's useful to color these components with red if their accessibility is REQURED or yellow if their accessibility is NOT-REQURED. In other cases, colors other than red and yellow can be used. The code-checking program color-codes a building component in the VRML model according to these rules. All VRML objects are hyperlinked to a set of comments in the Comments frame, because it's useful to know that an object complies with the building code.

The VRML components are generated, because they are linked to the Comments frame. The Comments frame is dynamically generated with the corresponding HTML anchors. The links in the Comments frame to the building code document frame have been predetermined for each encoded provision. The building code document already has the predefined anchors associated with the possible links that are generated in the Comments frame.

## 7. CONCLUSIONS

The purpose of this paper is to solve the design standards processing problem in one domain and then apply the principles of the code-checking framework to other domains. The principles behind the proposed approach of constructing integrated client/server framework for operative checking of building code developed in this work should be applicable to different aspects of the building code.

Exploration of the information contained in a building model is an important issue. Attributes, that are related to a building code should be generated by the code-checking program as opposed to being explicitly defined within the building model. If a designer is going to employ the services of a code-checking program, the program, not the designer, should make the determination of whether an exit path exists. Subsequent releases of the IFCs will define more code-related processes and it would be interesting to extend the approach to make it applicable to other areas of code checking.

## REFERENCES


De Waard, M. (1992). "Computer aided conformance checking: checking residential building designs against building regulations with the aid of computers." Ph.D. thesis, Technical Univ. Delft, Hague, The Netherlands.

Fenves, S. J., Garrett, J. H., Kilicote, H. (1995), "Computer representations of design standards and building codes: U.S. perspective." *The Int. J. of Constr. Information Technol*., University of Salford, Salford, U.K., 13-34.

"Industrial automation systems and integration – product data representation and exchange. I: Overview and fundamental principles." (1994). ISO 10303-1:1994, Tech. Comm.184, Subcomm. 4, International Standards Organization, Geneva, Switzerland.

*Industry foundation classes release 1.5, specification volumes 1-4*. (1998). International Alliance for Interoperability, Washington, D.C.

Kiliccote, H. (1996). " A standards processing framework." Ph.D. thesis, Dept. of Civil and Envir. Engrg., Carnegie Mellon University, Pittsburgh, Pa.

Yabuki, N. (1992). "An integrated framework for design standards processing." *Tech. Rep. 67*, Ctr, for Integrated Fac. Engrh., Stanford University, Stanford, Calif.