A CLASSIFICATION AND INDEXING SCHEME FOR CONCEPTUAL BUILDING DESIGN

Hugues Rivard, Steven J. Fenves, Nestor Gomez

ABSTRACT

SEED is a multidisciplinary project intended to provide support for the early phases of building design. SEED-Config is the module of SEED that focuses on the generation of the 3-dimensional configuration of spatial and physical building components. The architecture of SEED-Config consists of four software modules: a design information repository; a design knowledge manager; a classification reference manager; and a geometric modeler. The design information repository is built upon a generic information model, called the Building Entity and Technology (BENT) model, that stores design data as they are generated during conceptual design, supports case-based reasoning, and shares data among all design participants. The model represents each building entity as a generic container that encompasses its geometry, taxonomy, composition, relationships, and properties, as well as the design knowledge that was applied to it. This paper presents the classification and indexing scheme that has been developed on top of the generic information model in order to classify, access, and retrieve previous design solutions. The scheme uses a faceted classification to define the controlled vocabulary from which indexes are obtained. In this approach, classification is independent from the information model. The classification is extensible and designers have the freedom to complement the vocabulary with their own terms. Indexing is performed automatically as building entities are designed through the selection and application of technologies. Hence, a design is already indexed when it is added to the case library.

KEYWORDS: *conceptual building design, classification, indexing, case-based reasoning.*

1. INTRODUCTION

SEED (Software Environment to support the Early phases in building Design) is a multidisciplinary project intended to provide computer support for the preliminary design of buildings. The emphasis is on supporting early design exploration, that is, the fast generation of alternative design concepts and their rapid evaluation against a broad spectrum of relevant - and possibly conflicting - criteria [Flemming and Woodbury 1995]. The SEED project is subdivided into three main modules: SEED-Pro, which supports the generation of an architectural program; SEED-Layout, which supports the generation of schematic layouts; and SEED-Config, which supports the design of a three-dimensional building configuration in terms of spaces, subsystems, and physical components [Woodbury and Chang 1995]. This article focuses on the SEED-Config module.

Designers tackling new design problems typically refer to prior experiences. Thus, a computer tool that would store prior design experiences and make them available to designers could be of great help for seasoned as well as for novice designers. A design environment incorporating case-based reasoning provides such a tool. It could help designers recall previous and appropriate

cases which could be used as sources on the basis of which relevant solutions to the current problem may be developed. A typical design firm may easily handle dozens of building projects in a five year span, each project consisting of thousands of building entities. The collection of these building entities represents an invaluable repository of design knowledge. In order to facilitate the retrieval process and to maximize the value of such a collection two complementary tasks are necessary: classifying and indexing [Downing and Downing 1992]. Classifying involves the elaboration of a classification by arranging objects into categories and classes, while indexing is the process of assigning meaningful indexes to each of the objects for retrieval purposes. These two tasks are intertwined because classification deals with the organization of knowledge and provides a logical foundation for indexing.

This article describes a classification and indexing approach for conceptual building design. Before describing this approach, the article presents the architecture of the SEED-Config prototype followed by the building design representation and the information model adopted. Additional information may be found in [Rivard 1997].

2. ARCHITECTURE OF SEED-CONFIG

The SEED-Config prototype is divided into four application modules as shown in Figure 1: a design information repository; a design knowledge manager; a classification reference manager; and a geometric modeler. The design information repository records design data and manages design projects. It provides the slate upon which designs are created, edited, and browsed. It is also used to retrieve, archive, and browse cases in the case library. The design knowledge manager handles the collection of technologies selected by the designer that are to be used for the project; it allows the browsing, creation, editing, selection, and application of technologies which encapsulate design knowledge (technologies are described in Section 4). The classification reference manager is used to define, manage, infer about, and query classifications. The geometric modeler is used to define, reason about, and render both topology and geometry. Figure 1 depicts how the four modules interact with each other.

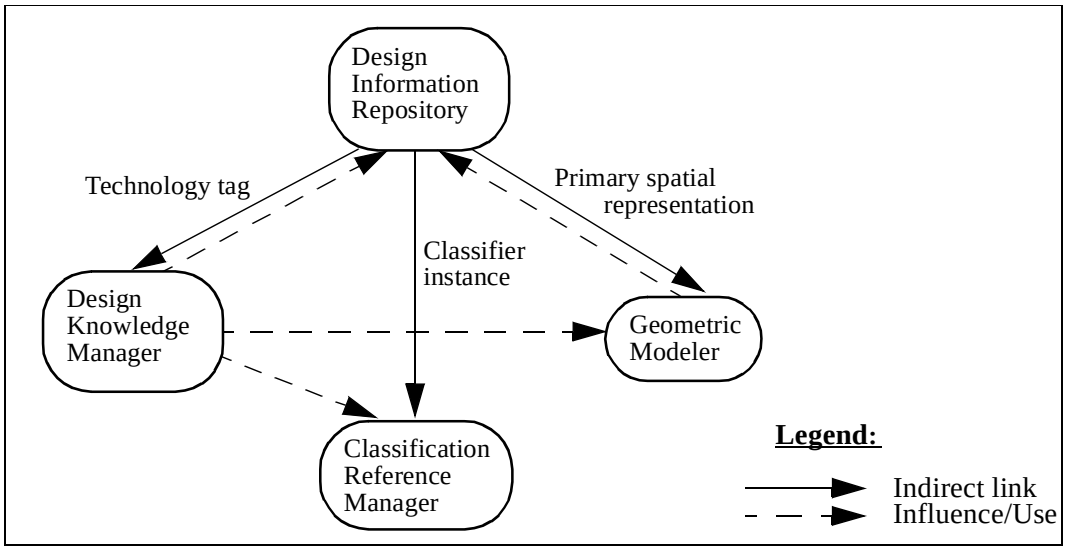

Figure 1. The four software modules of SEED-Config.

Each of the four application modules described above handles a specific type of document. The term document is used here in the context of an application framework. It is the central component of an application that encapsulates and manages the data model [Weinand et al. 1989]. For instance, the design information repository handles building decomposition hierarchy documents while the classification reference manager handles classification documents. A building design is considered complete and fully documented only if it references a classification, the design knowledge used, and a geometrical description. This is why a building decomposition hierarchy document keeps indirect links to data generated by the other three modules. An indirect link is implemented as an object that uniquely specifies a single element in the document of the target module (the names of these objects are shown beside their corresponding arrows in Figure 1).

This architecture results in a design environment that separates design representation, design knowledge, classification, and geometry. This is a radically different approach from the information models proposed in the literature where functionalities of the four modules are integrated (e.g., [Bjork 1989; Seren et al. 1993; Biederman and Grierson 1995; Rosenman and Gero 1996]). The data in each of the four modules are independent and can evolve independently from data in other modules. Hence, designers are free to augment or modify the system-provided conceptual models, classifications, and technologies independently. Furthermore, this modular architecture allows each module to be implemented independently, allowing concurrent development of the prototype.

3. MODELING BUILDING DESIGNS IN SEED-CONFIG

SEED-Config accounts for decisions on the overall form of the building and the structural, enclosure, mechanical, electrical, and interior systems [Woodbury and Chang 1995]. In order to support the various designers collaborating on a specific project, an information model that provides the basic data structure to support design process integration is needed. Design process integration is defined to be the continuous and interdisciplinary sharing of data, knowledge, and goals among the various designers [Fischer and Froese, 1996].

The strategy adopted is to define a generic information model shared by the various design domains involved. This information model defines a few basic constructs that completely capture the design of a building at the conceptual stage. These basic constructs can potentially describe all the aspects of a building such as the enclosure, the structure, or the heating, ventilation and air conditioning systems. The information model is specialized into a conceptual model (or schema) for each domain. The conceptual model defines, for a particular domain, the main objects, their relationships, and their data. Therefore, three levels of abstraction models are introduced: the data model, the information model, and the conceptual model. Figure 2 shows the three levels of abstraction models along with the ones selected or defined for SEED-Config. At the lowest level, the **data model** consists of a collection of object types, general integrity rules, and operators [Date 1995]. Two popular examples are the relational and the object-oriented models. The object-oriented model is selected here for its superior modeling capabilities. The **information model** is a set of basic constructs, built on top of a data model, enabling the organization of data about a product in a logical and structured manner. An information model for a product carries richer constructs than a data model and spans the product's life cycle [Eastman and Fereshetian 1994]; it deals with the syntax of the design information. The **conceptual model** of an object is

a computer-based representation of an object whose definitions depends on the observer's perception and conception [Rosenman and Gero 1996]. It refines an information model by specifying the categories of information used in a particular domain [Bjork 1992] and thus specifies the semantics of the design information for the domain.

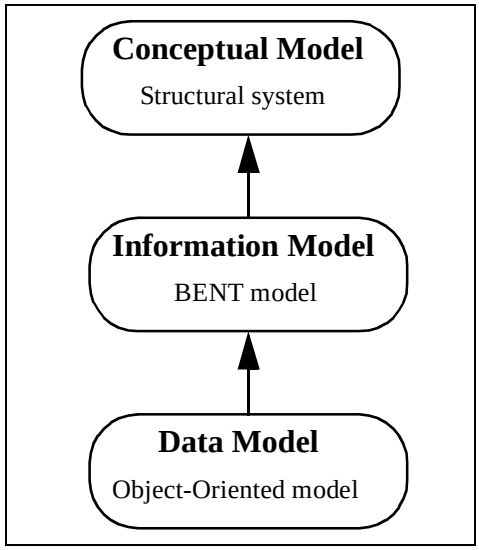

Figure 2. Three levels of abstraction model.

4. THE BUILDING ENTITY AND TECHNOLOGY INFORMATION MODEL

The design information repository is built upon a generic information model, called the Building Entity and Technology (BENT) model, that stores design data as they are generated during conceptual design, supports case-based reasoning, and shares data among all design participants. In the BENT model, the building is represented as an assembly of building entities with relationships among them. An entity is something that can be distinctly identified in a building design and about which data are accumulated [Eastman and Fereshetian 1994]. Each entity represents a concept meaningful to design participants, such as a beam, a room or a structural frame. An entity can be a system, a sub-system, a component, a part, a feature of a part, a space or a joint [Gielingh 1988]. All building entities are modeled with a generic container that glues together the modular information on an entity and provides a single standardized interface for accessing information.

A building entity records the name of the entity, the name of the author who created it and includes the following elements: components that organize attribute-value pairs (described below); classifier instances (indexes described in Section 6); relationships (containment and domain-specific relationships); technology tags, each of which refers to a technology node (described below); groups; and geometrical descriptions (primary spatial and secondary geometrical representations) [Rivard et al. 1996].

The attribute-value pairs of an entity are combined into small cohesive subsets, each of which is called a component. An advantage of grouping attributes into components is that closely related attributes are found together. For instance, section properties of structural members such as depth, area, and moment of inertia are found in the same component. Components allow entities to be refined in staged steps by adding sets of attribute-value pairs to the entity as they are generated in the design process. Hence, there is no need to predict all possible attribute-value pairs needed in a building entity at the outset. Components can also be used to present views to the user. Only the relevant components are displayed to the user, thus abstracting the complexity of the building entity. Data integration is supported by sharing components among different domains.

The BENT model separates design knowledge from design data. This partition allows the design knowledge to evolve independently from the stable information model. Technology nodes encapsulate design knowledge applied to building entities. A node represents a known design alternative, the constraints that determine its applicability, and the computational steps necessary to assign values to the attribute(s) defining that alternative. Technology nodes are organized into a hierarchically structured technology graph. This technology graph assists designers in designing a building entity by offering various alternatives. In the structural design domain, the technology graph represents the various alternative structural systems, subsystems and types of elements available to the designer. The root of a graph operates on an abstract building as a whole, while nodes at succeeding levels operate on more and more specific building elements. Hence, the technology graph may deal with elements ranging from the most abstract (e.g., a full 3-D building for which a tube structure may be an alternative structural system) to the most specific elements (e.g., individual beams or even connections, reinforcement, etc.).

5. CLASSIFICATION

Classification is a fundamental component of human thought. The words used to think and communicate are frequently names of classes. Designers perform numerous acts of classification every day. Is a structural element a column, a beam, or a truss member? Is a column short or slender? These classifications have major impacts on structural design and, thus, computerized design environments need to support the classification of design artifacts and components. This section describes the classification scheme adopted for SEED-Config.

5.1. A Faceted Classification Scheme

Classification schemes have been developed to provide a sound basis for classifications. SEED-Config uses the faceted classification scheme. In this scheme, an object is categorized by a set of facets, where each facet defines a specific aspect of the object. With each facet is associated a group of classes that are arranged hierarchically. An object is classified by selecting one class from one or more facets associated with that object. Faceted classification schemes have the advantage of being more specific than other schemes such as enumerative ones [Aluri et. al 1991].

In SEED-Config, a facet is called a category and a class is called a classifier. A **category** consists of a hierarchy of classifiers that characterize a specific aspect of an object. A **classifier** is a label assigned to a group of objects sharing a common characteristic. For instance, “timber” is a classifier that may be assigned to structural components made out of wood. Classifiers are arranged in a subsumption hierarchy. Subsumption is the process of recognizing a narrower concept (or classifier) to be part of a broader one. Hence, classifiers are more specific as one travels down and more general or abstract as one travels up the hierarchy. The classification hierarchy is built up by successively dividing a concept into more specific classifiers based on a single characteristic at each level of the hierarchy. An example of a category in structural design is the structural material category, which may specialize into four classifiers: steel, concrete, timber, and masonry. Figure 3 shows the “Structural material” category and its partial hierarchy of classifiers. Although most hierarchies of classifiers investigated have been found to be strict trees, some notable exceptions have arisen. Hence, a classifier can have more than one generalization within a given category. Figure 4 shows an example of a category with two classifiers having multiple generalizations. In this example, the “composite steel and concrete” type of decking as well as

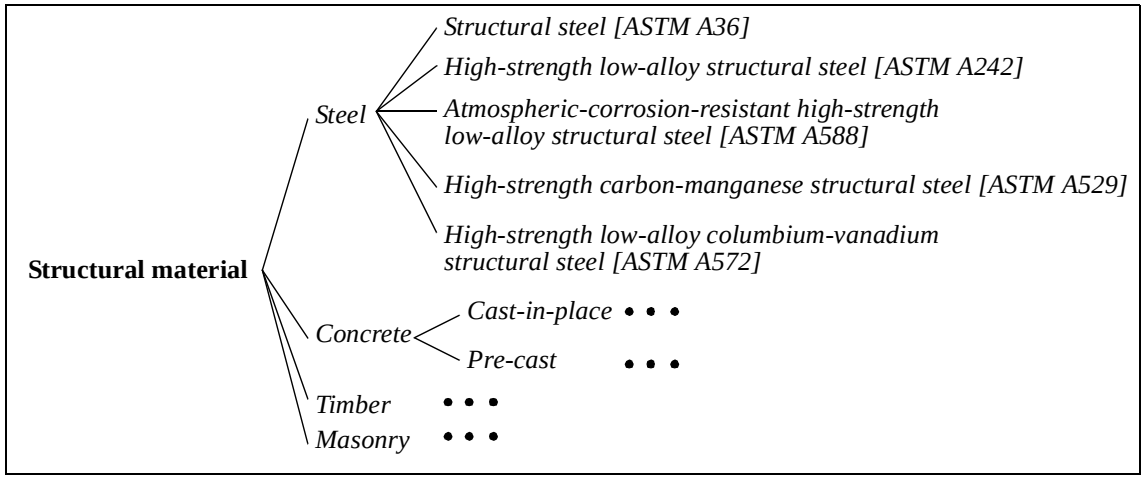

Figure 3. A category and its partial hierarchy of classifiers.

the "corrugated steel" type of decking can both be specialized into either "cellular" or "non-cellular" decking. An efficient classification is obtained when all classifiers at a given level of the hierarchy are mutually exclusive and collectively exhaustive (closed world assumption) [Langridge 1992]. These two requirements apply to classifiers having either single or multiple generalizations. It may be difficult to satisfy these requirements in every situation, but it appears that an adequate classification can still be obtained if these are relaxed [Stasiak 1996].

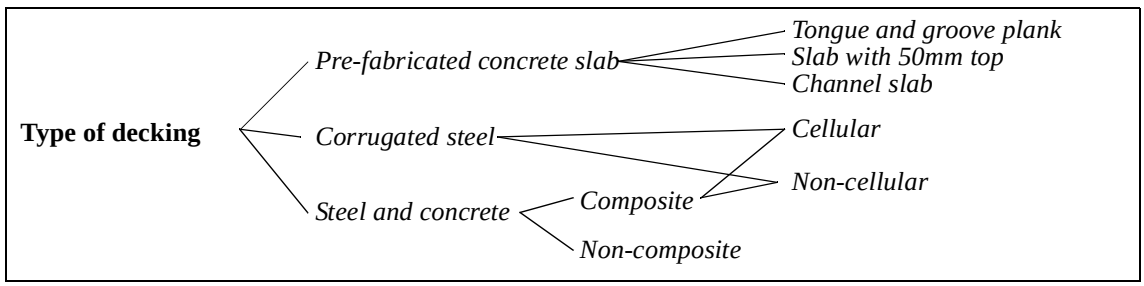

Figure 4. An example of classifiers having multiple generalizations.

Building entity type is a special type of classifier used to categorize each generic building entity as one of the nodes of the hierarchical decomposition defined by the conceptual model. Hence, a specific structural frame is represented as a building entity classified with the building entity type "frame". A particular building entity type may have a number of categories associated with it. The set of categories differs from one building entity type to another. Figure 5 shows three building entity types along with their categories and partial hierarchy of classifiers. The example shows that a category, here "structural material", can be shared by more than one building entity type. A building entity can be classified by an arbitrary number of classifiers, but with at most one classifier from each of the categories associated to its assigned building entity type(s). This restriction arises from the requirement that classifiers within a category be mutually exclusive at a given level of the classifier hierarchy. It is important to note that the scope of a category is limited to its associated building entity type(s). Hence, the category "Type of lateral structure", in the example, is limited to the "Vertical lateral subsystem" type and is not inherited by the "3D-System" or "Frame" types.

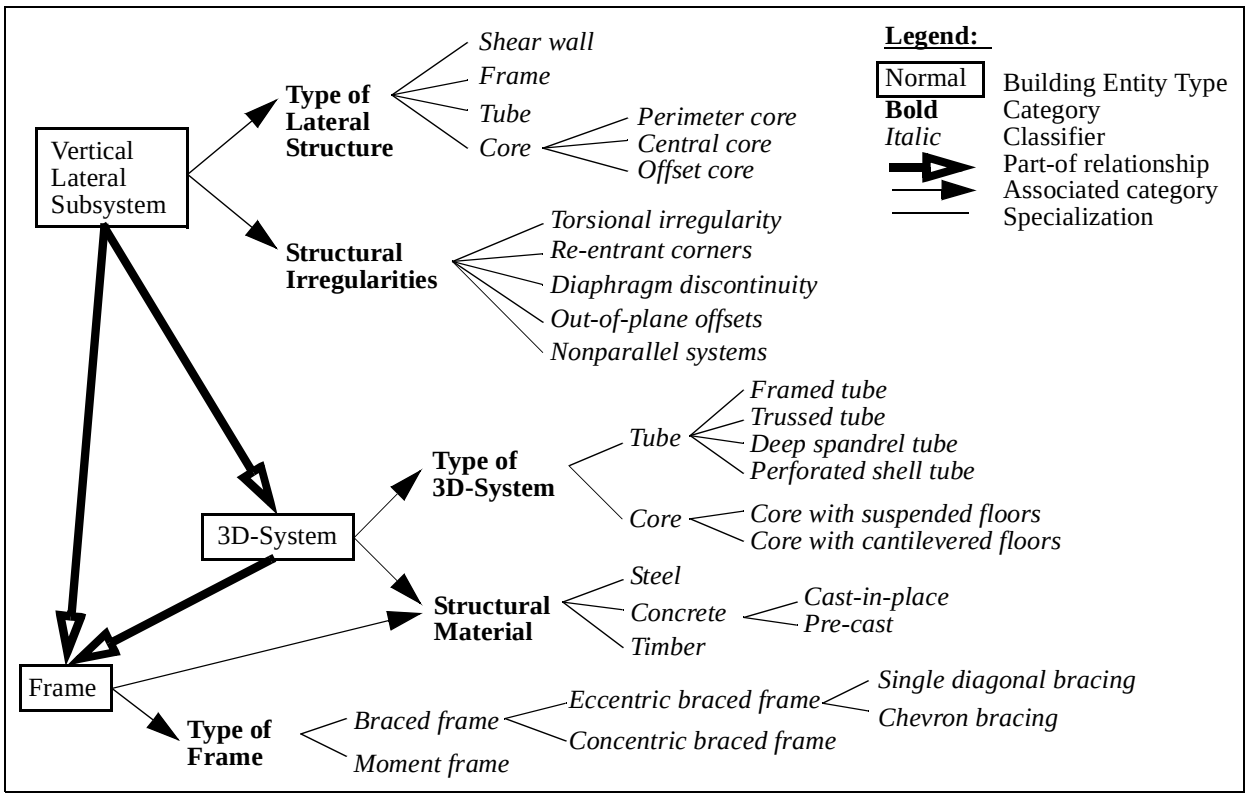

Figure 6. Three building entity types, their categories and their partial hierarchy of classifiers.

When a building entity performs several functions, more than one building entity type can be assigned to it. For instance, a building entity that acts as a load-bearing exterior wall would be assigned two building entity types: a “structural wall” and an “envelope plane”. This capability provides support for integrating different design domains. The same generic entity can be classified according to two different design domains through the assignment of classifiers pertaining to categories of the two domains.

5.2. Classification Documents and Conceptual Models

Each design project in SEED-Config has associated with it a classification document. A **classification document** contains the classification used in the design of that project. It defines the domain of all legal categories and classifiers. The classification document is used for preparing queries to retrieve design cases and is also used by the technologies when refining or elaborating a building entity. The classification document is an important part of a conceptual model since it defines the hierarchical decompositions and the classifiers used in a design. It defines the hierarchical decomposition by outlining the building entity types of the design domain, their associated categories and the corresponding classification hierarchies. Like the technology construct which abstracts the design knowledge away from the information model, the classification document separates the taxonomy from the information model. This allows the classification to evolve and change without affecting the information model.

Because the classification scheme is separated from the information model, different classifications can be supported with the same infrastructure. This is important since a number of equally

valid classifications can be generated for a given domain. This capability has the additional advantage of allowing a number of classifications (or conceptual models) for different design domains to be associated with a given building design project. For instance, a project could be associated with a structural classification for structural design, with an enclosure classification for enclosure design, and so on.

To avoid the mismatch problem that arises from having different classifications for a given domain, a unique classification should ideally be developed for each domain. Categories and classifiers from such a classification should mirror the terminology used by designers in order to act as a controlled vocabulary. Since this controlled vocabulary represents the basis for both indexing and retrieval, it should alleviate the typical retrieval problem of vocabulary differences between queries and documents [Stasiak 1996].

6. INDEXING

An index is a pointer (or an indicator) to which a keyword (or label) is assigned and which leads to information about a specific and related topic in a large collection of data. The purpose of an index is to facilitate the retrieval of specific information from a large repository. A common example is a book index which consists of a list of tuples (i.e., keywords and page numbers) pointing to specific locations in a book. Since a complete building design represents a large quantity of information, the use of indexes in design facilitates the retrieval of relevant design data. Actually, indexes in design can perform two useful functions: to facilitate the retrieval of specific information in an on-going design project, and to allow the retrieval of past design solutions in a library of cases. This section describes the indexing scheme used in SEED-Config and its role in case-based reasoning.

6.1. Indexing Scheme

A typical problem faced in index assignment is what terminology to use (e.g., are you designing a cinema, a movie theater, or a motion-picture theater?). Furthermore, for an index to be meaningful, it must be understood by and be obvious to most users. To avoid such indexing problems, keywords need to be selected from a controlled vocabulary. Such a vocabulary ensures that indexes will be assigned in a consistent manner so that information about the same subject will be found with the same keyword [Downing and Downing 1992]. Classification schemes are frequently used as the basis for indexing. For instance, a widely used numerical classification scheme for data filing, construction specifications, and estimates is the 16 division format: (1) general data, (2) sitework, (3) concrete, (4) masonry, and so on. This 16 division format has been adopted by the American Institute of Architects (MASTERSPEC), the Construction Specification Institute (SPECTEXT), and Sweet's Catalog [Sweet's 1994], among others. Using a classification scheme as the controlled vocabulary for indexing has the additional advantage of producing index that are arranged in a systematic and logical order [Langridge 1992]. Hence, in order to avoid terminology problems and to have indexes organized in a systematic and logical order, SEED-Config relies on the classification scheme presented in the previous section as the controlled vocabulary.

Indexing in SEED-Config is done using classifier instances. A **classifier instance** is a label assigned to a building entity for the purpose of classifying and indexing it. Each classifier

instance corresponds to only one classifier from the classification document. A classifier instance is a distinct construct that keeps a reference to the classifier it pertains to and that contains additional information relating to the design (i.e., references to the author and the technology that assigned the classifier instance). Whenever a classifier instance is assigned to a building entity, the generic entity construct becomes a member of the class defined by the corresponding classifier. Hence, classifier instances have two important purposes: they categorize the generic building entities into domains and sub-domains as they are being refined during the design process; and they act as indexes in case-based retrievals and in queries for data from the on-going design. The advantage of classifier instances is that they provide access to building entities and design information independently from the design sequence that generated them. A given building entity may have several classifier instances assigned to it, each of which offers a different handle to retrieve it. Hence, a search can be based on a unique classifier or on the conjunction of a set of classifiers. Figure 6 shows how a building entity is assigned a set of classifier instances based on the associated classification. A classifier instance can be an instance of either a building entity type or a classifier. A building entity may correspond to more than one building entity type in the case where it has multiple functions (e.g., a load-bearing enclosure wall). Classifiers must be selected from one of the categories associated with each of the building entity types. Only one classifier from each category can be applied to the building entity.

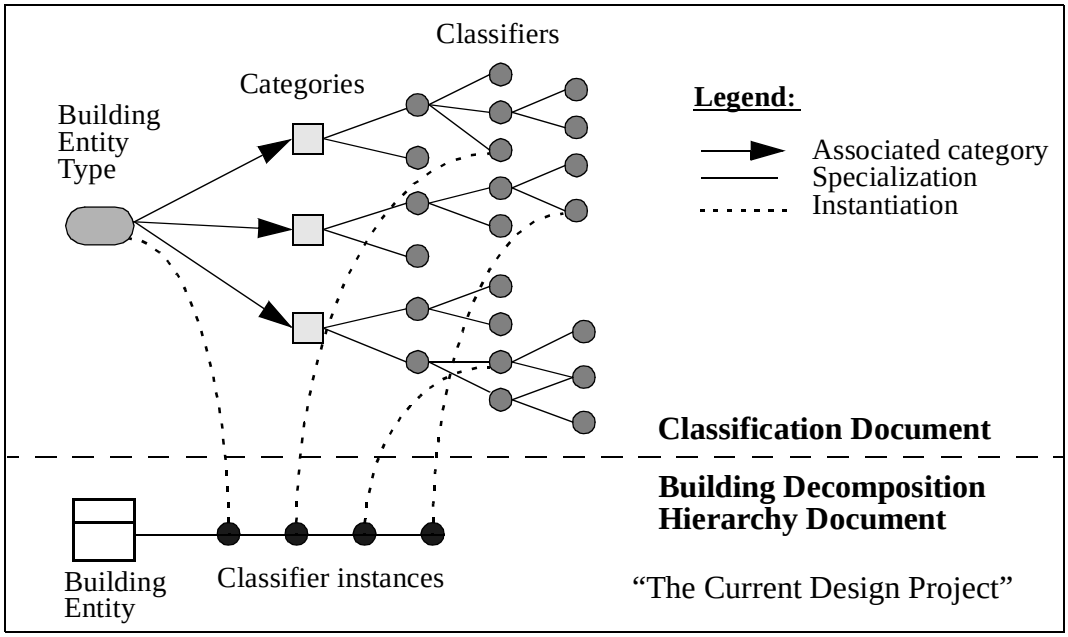

Figure 6. Classification and indexing of a building entity.

A classifier instance is assigned to a building entity whenever the value of an attribute is a symbolic value selected from a predefined set, such as the type of structural materials. Indexes must be assigned to design data before a query is attempted and to a case before it is added to the case library. The index assignment process is concerned with selecting tags that differentiate the case (or the design) from others and that indicate the situations when the case (or the design) could be useful in solving a problem [Kolodner 1993]. SEED-Config requires an efficient and automatic indexing mechanism because of the size envisioned for the case library and to relieve designers from the indexing chore. This is why indexing in SEED-Config is performed automatically as

designers select and apply technologies to building entities. Technology nodes assign classifier instances automatically according to the design knowledge they encapsulate. Hence, a design is automatically query-ready and a case is already indexed when it is added to the case library.

6.2. Indexing and Case Retrieval

SEED-Config assigns indexes (i.e, classifier instances) to cases during the design process and before the cases are inserted into the case library. During retrieval, these indexes are used to find the most appropriate cases for solving the current problem. Indexes are in fact a subset of the case representation and thus, need to record problem descriptions, solutions, and outcomes. Indexes ensure that searches are executed efficiently and flexibly [Kolodner 1993].

SEED-Config's classification scheme is hierarchical in nature and thus provides an efficient organizational structure for searching cases because only a subset of the case library is examined. Classification hierarchies are in fact semantic networks, which provide an efficient mechanism for supporting reasoning about the generalization or specialization of concepts. This is an essential capability for widening the space of applicable cases when a query results in a limited number of matches. Say, for instance, that a structural engineer is interested in designing the structure covering a swimming pool facility and requests help from the case-based reasoner. An initial query can be specified as: "Type of building entity = Functional space" and "Occupancy = Swimming pool". Note that the category "Occupancy", shown in Figure 7, is associated with the building entity type "Functional space". If the query is unsuccessful because there are no such cases in the case library, the query can be generalized, through the semantic network (i.e., classification hierarchy), to sport facilities or even to halls in order to possibly find the structural design of an auditorium that could be reused. Hence, a second, more general, query can be specified as: "Type of building entity = Functional space" and "Occupancy = Hall" which subsumes "Sport facility", "Entertainment", and their respective specializations.

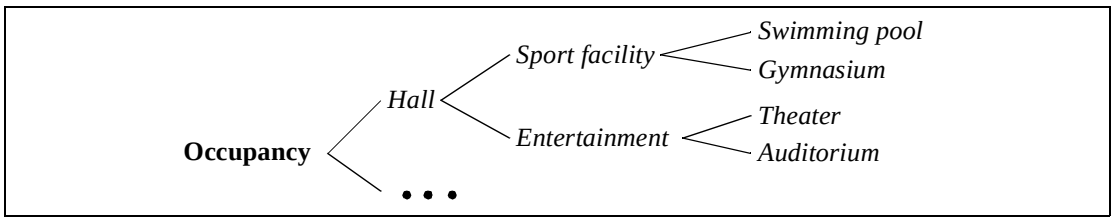

Figure 7. The "occupancy" category and its partial classification hierarchy.

7. THE CLASSIFIER APPROACH VERSUS INHERITANCE FOR CLASSIFICATION

The use of class inheritance for classifying objects has led to many excesses in object-oriented information models. Inheritance should be used for software reuse only and not for classification purposes, as is the case in some information models found in the literature (see [Biedermann and Grierson 1995] or [Ekholm 1996] for example). Misuse of class inheritance has resulted in problems such as non-homogeneous characterization hierarchies, the exponential explosion of number of classes, the incorrect use of specialization over aggregation in certain situations, and the complexity of constructing and maintaining an inheritance hierarchy [Howard et al. 1992; Kiliccote 1992]. The main advantage of the classifier approach adopted in this study is that classification is independent from the information model. This advantage ensures the consistency of the informa-

tion model at all times. It allows the ad hoc expansion of classifier hierarchies without affecting the information model. It also permits an object in the information model to be classified into more than one type of entity. Hence, a given building entity can be classified as both an envelope plane and a structural wall. Another advantage of having classification orthogonal to the information model is that multiple inheritance, with its conflict problems, is avoided in the information model but supported in the classification scheme. Furthermore, having a specialized tool such as the classification reference manager to deal with classification provides the support for complex reasoning that a normal inheritance class hierarchy cannot provide. It supports "is a" queries (e.g., is this building entity a beam?), queries involving subsumption inference (e.g., is a purlin a beam?), and class partitioning (e.g., a column can be either short, intermediate, or slender).

8. CONCLUSION

Since classification is an intrinsic part of design, SEED-Config associates a classification document with each design project. This document contains a faceted classification that defines the domain of all legal classifiers for a given design domain. It defines the model of the hierarchical decomposition using building entity types, each of which has a set of categories with related hierarchies of classifiers. The hierarchies of classifiers act as semantic networks that support generalization or specialization of searches. An application called the classification reference manager is provided to allow designers to create, modify, or augment their own classifications. This provides designers with the freedom to complement the terminology used in their design domains by incorporating their own terms.

The classification associated with a design project is used as a controlled vocabulary from which indexes are obtained. The controlled vocabulary ensures consistent index assignments and avoids terminology problems. Indexing is performed automatically as designers select and apply technologies to building entities. Hence, a case is already indexed when it is added to the case library. Such an automatic indexing is a necessity due to the size of cases representing buildings. No other design environment found either in industry or in the literature offers such a capability to classify and index design data.

9. REFERENCES

Aluri, Rao, D. Alasdair Kemp, and John J. Boll (1991). *Subject Analysis in Online Catalogs*, Libraries Unlimited, Inc., Englewood.

Biedermann, J. D., and D. E. Grierson (1995). "A Generic Model for Building Design." Engineering with Computers, Vol. 11, No. 3, pp. 173-184.

Bjork, Bo-Christer (1989). "Basic Structure of a Proposed Building Product Model." Computer-Aided Design, Butterworth & Co. Ltd., Vol. 21, No. 2, pp. 71-78.

Bjork, Bo-Christer (1992). "A Unified Approach for Modelling Construction Information." Building and Environment, Vol. 27, No.2, pp. 173-194.

Date, C. J. (1995). *An Introduction to Database Systems*, Sixth Edition, Addison-Wesley Pub. Co., Reading.

Downing, Mildred H. and David H. Downing (1992). *Introduction to Cataloging and Classification*, Sixth Edition, McFarland & Company, Inc., Jefferson, NC.

Eastman, Charles M., and Nirva Fereshetian (1994). "Information models for use in product design: a comparison." Computer-Aided Design, Vol. 26, No. 7, pp. 551-572.

Ekholm, Anders (1996). “A Conceptual Framework for Classification of Construction Works.” ITcon, Vol. 1, [web page], accessed on March 1996, http://itcon.org/

Fischer, Martin, and Thomas Froese (1996). “Examples and Characteristics of Shared Project Models.” Journal of Computing in Civil Engineering, ASCE, Vol. 10, No. 3, pp. 174-182.

Flemming, Ulrich, and Robert Woodbury (1995). “Software Environment to Support Early Phases in Building Design (SEED): Overview.” Journal of Architectural Engineering, ASCE, Vol.1, No. 4, pp. 147-152.

Gielingh, W. (1988). “General AEC Reference Model.” ISO TC 184/SC4/WG1 doc 3.2.2.1, TNO Report BI-88-150.

Howard, H. Craig, Jamal A. Abdalla, and D. H. Douglas Phan (1992). “Primitive-Composite Approach for Structural Data Modeling.” Journal of Computing in Civil Engineering, ASCE, Vol. 6, No. 1, pp. 19-40.

Kiliccote, Han (1992). “The Context-Oriented Model: A Hybrid Approach to Modeling and Processing Design Standards.” M. S. thesis, Department of Civil and Environmental Engineering, Carnegie Mellon University, Pittsburgh, PA.

Kolodner, Janet L. (1993). *Case-Based Reasoning*, Morgan Kaufmann Pub. Co., San Mateo.

Langridge, Derek W. (1992). *Classification: Its Kinds, Systems, Elements and Applications*, Bowker-Saur, London.

Rivard, Hugues, Nestor Gomez, and Steven J. Fenves (1996). “An Information Model for the Preliminary Design of Buildings”, Proceedings of the CIB W78/TG10 Workshop “Construction on the Information Highway”, CIB Proceedings, Publication 198, Edited by Ziga Turk, Bled, Slovenia, June 10-12, pp. 435-448.

Rivard, Hugues (1997). “A Building Design Representation for Conceptual Design and Case-Based Reasoning.” Ph.D. thesis, Department of Civil and Environmental Engineering, Carnegie Mellon University, Pittsburgh, PA.

Rosenman, Michael A., and John S. Gero (1996). “Modelling multiple views of design objects in a collaborative CAD environment.” Computer-Aided Design, Vol. 28, No. 3, pp. 193-205.

Seren, Karl-Johan, Matti Hannus, Kari Karstila, Markku Kihlman, and Jukka Pellosniemi (1993). “Object-Oriented CAD Tool Implementations for the Construction Industry.” VTT Research Notes 1460, VTT Technical Research Centre of Finland, Espoo, Finland, 90 pages.

Stasiak, Dana M. (1996). “An Environmental Regulations Broker: Classification and Co-Word Analysis.” M.S. Thesis, Department of Civil and Environmental Engineering, Carnegie Mellon University.

Sweet’s (1994). “Engineering and Retrofit: Mechanical, Electrical, Civil/Structural.”, 16 Volumes, McGraw-Hill Inc., New York.

Weinand, Andre, Erich Gamma, and Rudolf Marty (1989). “Design and Implementation of ET++, a Seamless Object-Oriented Application Framework.” Structured Programming, Springer-Verlag Inc., New-York, Vol. 10, No. 2.

Woodbury, Robert, and Teng-Wen Chang (1995). “Massing and Enclosure Design with SEED-Config.” Journal of Architectural Engineering, American Society of Civil Engineers, Vol. 1, No. 4, pp. 170-178.