KNOWLEDGE DISCOVERY IN ARCHITECTURAL CAD DATA

GREGORY J SMITH AND MARY LOU MAHER
Key Centre of Design Computing and Cognition
University of Sydney
{g_smith,mary}@arch.usyd.edu.au
SIMEON J SIMOFF
Department of Computer Systems
University of Technology, Sydney
simeon@it.uts.edu.au

CONTACT AUTHOR:
MARY LOU MAHER
Key Centre Of Design Computing And Cognition
Bldg G04
University Of Sydney
Nsw 2006 Australia
Fax: 011 612 9351 3031
Phone: 011 612 9351 4108
email: mary@arch.usyd.edu.au

Manuscript:
20 pages
8 Figures
1 Table

KNOWLEDGE DISCOVERY IN ARCHITECTURAL CAD DATA

Abstract.
Knowledge discovery and data mining techniques have the potential to find patterns in very large sets of data. Applying these techniques to design data has two problems: the volume and complexity of the source data, and the discovered knowledge itself. A result is that existing algorithms may not be computationally feasible and in any case may not produce, of themselves, results that are of interest to a designer. In this paper we first demonstrate the two problems above by applying data mining techniques to architectural CAD data. We then reconsider the role of data mining for knowledge discovery as facilitating the re-presentation of source design data in other forms such that relationships are revealed to the designer that would not otherwise be visible. To investigate this we adopted a framework for knowledge discovery combining consecutive complementary strategies, and applied it to the same architectural CAD data.

Keywords: data mining, CAD data, knowledge discovery, design

1. Digital design representations and data mining

Computational support and digital design representations serve different purposes during different phases of a design process. Consequently, it is not surprising that digital design representations span the whole variety of today's multimedia computing, as shown in Figure 1.

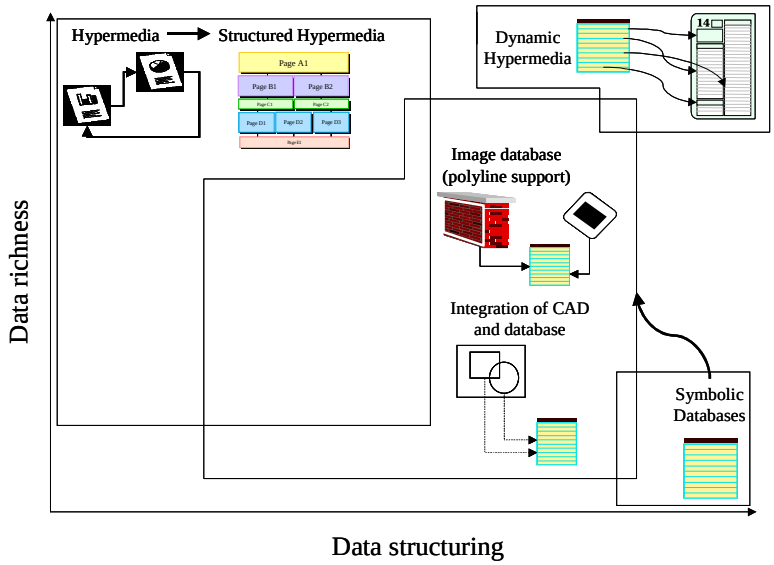

Figure 1. The scope of design data from a data mining point of view

Different forms and layers in digital design media vary with respect to structuring and richness of the design data that they hold. Design domain knowledge is coded in a variety of machine readable forms as electronic manuals and references, project databases, CAD drawings and database, images and 3D models, design communication transcripts and activity records.

- structure-valued data, including: (i) attribute-value pairs; (ii) relational tables; (iii) object-oriented data structures;

- weakly structured data, including: (i) texts in free, table format or structured communication records; (iii) vector graphics, such as CAD drawings, object-oriented images;

- raw data, including: (i) raster images of photographs, sketches; (ii) animated images; (iii) audio and video data;

- links data, including: (i) hyperlinks within weakly structured data; (ii) links between structure-valued components and elements of weakly structured data; (iii) links within structure-valued data; (iv) information about the sequence of visited links.

Data mining as a process (Fayyad, Piatetsky-Shapiro & Smyth 1996) is part of what is called

knowledge discovery - an interactive and iterative process of discovering useful knowledge from the patterns extracted by the data mining techniques. The knowledge discovery process consists of three main phases: 1) pre-processing, 2) data mining, and 3) post-processing. Data mining, the central phase, involves the application of statistical and machine learning algorithms with which to extract patterns in data. However, successful knowledge discovery requires that significant amounts of effort must be applied to the other phases.

The pre-processing phase deals with data preparation. Techniques of sampling, feature selection and feature construction deal with problems in data quality such as unfilled database fields or records, attribute noise and errors. In our case errors extend to the area of text errors, missing images or drawings, errors in the transcripts of design communications. The most obvious part of pre-processing is in transforming raw data into whatever format is required by the data mining algorithms. In design knowledge discovery tasks this is an extremely important stage, due to the variety of the data formats (usually non-table), and the necessity to deploy a wide variety of data mining techniques to these data. High data volumes and data dimensionality is a feature of knowledge discovery tasks that are often not seen by the designers of the underlying machine learning and statistical algorithms.

In the post-processing stage the data is interpreted and visualised. This has a feedback effect on data mining algorithm selection and application. For example, do the results need to be human interpretable or not? If yes then a decision tree may be selected, if no then a neural network may be selected. Post-processing is also concerned with evaluation: are the results statistically significant and reliable? Are they interesting with respect to the goals of the project?

The importance of the pre-processing and post-processing phases for knowledge discovery from design data is underlined by the nature of design “knowledge”. What exactly is denoted by design knowledge and how it is represented are major issues in design computation research. Financial and business data are usually well-structured in object-oriented or relational attribute-value representations. Producing a knowledge source to be the input to a data mining algorithm is roughly equivalent to de-normalising a database schema into a flat table. The rows of such a table are design cases and data mining algorithms look for combinations of repetitions in attribute values, as shown for example in Figure 2. Such a representation makes almost explicit the associations between different attributes of the case.

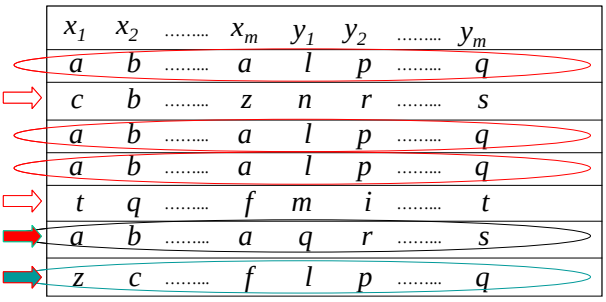

Figure 2. “Flattened” database as the source for data mining

It would be convenient if such an approach to the composition of knowledge sources worked in the mining of design data too. However, the content of the table cells would be heterogeneous as the source data includes drawings, product descriptions, 3D models, design communications, and project management data. What exactly should be mined, and how we should combine and utilize the outcomes of different data mining algorithms performed over various components of design data, are questions that do not have trivial answers.

Until the early 1990's, rule-based representations were amongst the most popular formalisms in artificial intelligence as they were seen to be "understandable" both by humans and computers. They were naturally adopted into a field of data mining that at that time was in its early days. Rule-based representations can be used for further reasoning, explanation of the discoveries, formulating goals and reasoning chains. Some other representation schemata may have computational advantages but can be difficult for humans to understand. A trained neural network, to take an example, is capable of computing output values from the input values that it is supplied with, but it can not provide any meaningful explanation of how it is computing those values.

So, we have two problems regarding design data which make their application to data mining and knowledge discovery difficult compared to the domains normally considered: the volume and complexity of the source data, and the discovered knowledge itself. A result is that existing algorithms may not be computationally feasible and in any case may not produce, of themselves, results that are of interest to a designer. The approach we adopt has origins in the field of psychology. Von Glasersfeld, describing Piaget, distinguished representation from "re-presentation" so as to emphasize the construction of meaning of an object. The process of re-presentation in humans takes items which are considered given and coordinates them as content into a new form, with the resulting products being available as given in some future process (von Glasersfeld 1995).

The approach canvassed in this work, then, is to view our goal of knowledge discovery on design data as facilitating the re-presentation of source design data in other forms such that relationships are revealed to the designer that would not otherwise be visible. To investigate this we adopted a framework for knowledge discovery combining consecutive complementary strategies, and applied it to a real world design data (Simoff and Maher, 1999). The strategies were:

1. *Data-driven exploration*, where we do not specify what we are looking for before starting to examine the case data. The result of this step is building of an initial set of patterns that extend a priori knowledge and allow a refinement of the goals of the knowledge discovery process. For example, in the text analysis of a case library of design descriptions we started with finding the frequencies of word occurrences in the text. As a result we build an initial vocabulary – the initial set of semantic patterns that can be refined to become the basis for the formal semantic models for indexing and adapting the design cases (Simoff and Maher, 1998b).

2. *Hypothesis- or goal-driven exploration*, where we initially specify what we are looking for (i.e. we formulate a hypothesis), which is either refined during the exploration, partially or completely reformulated or finally rejected. This corresponds to the classical machine learning approach. The hypothesis for this stage may come from domain knowledge or from the output of the data-driven exploration stage. For example, we specify ontology based on the initial vocabulary and the information model of the design data. This can be viewed as a hypothesis for a classifier tree in the domain. Then the ontology is modified according to derived co-occurrences, concordances and the results of correspondence analysis of the text in the cases.

2. Knowledge sources and strategies for data mining

In our previous work on design data mining (Maher and Simoff, 1998; Simoff and Maher, 2000) we considered two types of source data: a hypermedia case library of buildings (http://www.arch.usyd.edu.au/kcdc/caut) and online conversations in collaborative design sessions. In both of these sources, the data was treated as text and the data mining algorithms relied on various

text analysis approaches. These two data sources were developed in the University as part of our teaching and research in computer supported design.

In the current study we looked at the data sources developed in an architectural practice. Specifically, we considered a single architectural project, with data taken from the following sources:

1. ArchiCAD archive of a hospital.
2. Briefing database, containing requirements that lead to the above CAD archive.

In this work the CAD data was the target of the data mining algorithms, with the briefing database being used to generate the room names and categories from which class labels were generated. The entire CAD archive and the briefing database were too large to consider, so we reduced the data to information about the rooms including the type, the furniture and equipment and any attributes that were stored in the CAD database. We extracted the relevant data from the CAD archive and put it in a relational database and then used SQL queries to obtain the following data:

1. List of Hospital Planning Unit (HPU) codes, such as for “emergency care”, “surgical ward”, “medical ward”, “maternity” and “catering”.
2. List of standard furniture and equipment.
3. List of the attributes for each required room. Each row corresponded to one room and contained attributes such as name, code, location, dimensions and equipment.

ArchiCAD models are constructed using objects like walls and doors that are conceptually familiar to architects. Zones with position, size and a contained set of objects denote rooms. The implication of this for knowledge discovery is that these conceptual objects already exist in the data and thus are not themselves a useful goal. This contrasts with pattern matching and data mining techniques on vector images where discovering the conceptual objects is the goal (see, for example, (Nagy 2000)).

In ArchiCAD, the standard zone list contains the attributes shown in Table 1 for each room, with a second zone list containing the contents of each room.

Table 1. Attributes of a zone or “room” in ArchiCAD

Attribute	Type
ZoneStoryName	String
ZoneName	String
ZoneHeight	Numeric
ZonePerimiter	Numeric
SurroundWallSurface	Numeric
MeasuredArea	Numeric
TotalDoorsSurface	Numeric
TotalWindowsSurface	Numeric
ZoneCategoryCode	Numeric
ZoneCategoryName	String
WallsAlongPerimeter	Numeric
TotalDoorsWidth	Numeric
TotalDoorsWidth	Numeric
NofAllCorners	Numeric

NofConcaveCorners	numeric

The attribute names should be self-explanatory except perhaps for ZoneCategoryCode and ZoneCategoryName. A zone is an aggregation of spatial regions that have a similar function, and zone categories are the assignment of zones to functional categories chosen by the designer. Examples of zone categories include “room”, “circulation”, “office” and “plant”.

Three strategies were followed. The first was a data driven exploration of the data set. A data set was obtained from the CAD data and was pre-processed into the form required by the data mining package. Various algorithms were then run in an attempt to investigate the structure of the data and to uncover any underlying patterns in the data.

The second strategy was a goal driven one. The two goals used were to attempt to determine data patterns based on room topology and room content. Data sets describing the position of rooms in the hospital and the contents of rooms were obtained and the algorithms run again. The class label used on supervised algorithms was room type and was derived from the briefing database.

Finally, we decided to customize a technique based on the Activity/Space model (Maher, Simoff, and Mitchell, 1997). This is the “re-presentation” view of knowledge representation that was introduced earlier. The is to enable the presentation of the design to the designer in a different form such that patterns and relationships that may not have been apparent in the CAD data become so. Rather than our computational algorithms attempting to present mined knowledge as a finished product, we divide “knowledge discovery” into (i) classification and visualization tasks, and (ii) interpretation tasks. The first of these then is computational and is based on the Activity/Space Ontology. The latter relies on a designer and their domain knowledge.

The data mining algorithms were run under WEKA (Frank et al. 2000) on a Linux platform. As the data originated from MS Windows applications, platform independent representations that were easily manipulatable were required. This included XML dumps of the SQL queries and vertical bar delimited ASCII dumps of the ArchiCAD listings. Data cleaning and data transformation, from these formats into the ARFF format required by WEKA, used standard Linux/Unix utilities along with Java XML and XSL tools from Apache (The Apache Software Foundation 2000).

3. An Exploration of the Data

3.1 THE NATURE OF OUR DATA

One of the key results from computational learning theory is that of Vapnik-Chervonenkis dimension. Consider learning a classifier $f:S \rightarrow H$ from a sample space S to an hypothesis space H. The VC dimension is the largest set of examples that can be completely fit by H (Shavlik and Dietterich 1990), where H completely fits a set of samples if there is always an $h \in H$ consistent with any class labeling from the samples. If the number of training samples substantially exceeds the VC dimension then it is likely that any hypothesis $h \in H$ consistent with the examples is actually a good approximation of f. If the number is less than the VC dimension then f is unlikely to classify new samples correctly. For example, the VC dimension of axis-parallel rectangles in a d dimensional real space is 2d (Blumer et al. 1989).

Haussler (1988) uses this and Valiant’s learning framework (so called PAC learning), to construct results of the minimum number of independent random examples of a target concept on an instance space S defined by N attributes. Take as example hypotheses those that are simple conjunctions (in

conjunctive normal form, or CNF). If we let, for example, the accuracy and confidence bounds both be 0.05 then of the order of O(425 + 130N) independent random samples of the target concept are required such that, with probability at least 0.95, we can either produce a CNF hypothesis with at most error 0.05 or be told that the hypothesis is not CNF. What this means is that even when mining for very simple concepts, the number of sample instances necessary increases dramatically with the number of attributes. **This is why data sets normally used in data mining are "tall and thin"**, i.e. have lots of instances and few attributes.

In our CAD data this is not the case, and even the few attributes described in the previous section were either numeric or strings. Depending on the target concept we may not be able to obtain enough instances from within a single architectural project **– our data is better characterized as being "short and fat"**.

The obvious response to this is to look for more data from other architectural projects, which in our study would be from other hospital designs. However underlying learning theory is a requirement that samples be taken from the same probability distribution. Intuitively this requirement means that we should take samples corresponding to the same concept. To do otherwise may result in a classifier that is statistically valid but otherwise of dubious meaning. Thus to sample across two architectural designs A and B so as to determine stylistic patterns, say, would require that A and B were equivalent in terms of size, scale, functionality, style and so on. Even assuming that such knowledge was available from design experts, using that knowledge to justify sampling across designs so as to uncover patterns across designs would be a case of circular reasoning.

3.2 ASSOCIATION RULE MINING

As a first pass at knowledge discovery involved inducing associations we ran the Apriori association rule learning algorithm on the ArchiCAD data set prepared as described in section 2. As Apriori requires nominal attributes, a minimum description length discretisation filter was used as a preprocessing stage. The zone category was used as a class label on this filter as the basis for transforming continuous attributes into a set of intervals. The set of intervals then served as an enumeration for Apriori resulting in large numbers of rules such as these:

$$\begin{aligned} &\text{(confidence=100\%)} \\ &\text{ZoneHeight} = \text{All} \wedge \\ &\text{TotalWindowsWidth} \leq 0 \quad \Rightarrow \quad \text{TotalWindowsSurface} \leq 0 \end{aligned}$$

$$\begin{aligned} &\text{(confidence=98\%)} \\ &\text{ZoneStoryName} = \text{“Level 1”} \wedge \\ &5.32 < \text{MeasuredArea} \leq 23.18 \wedge \\ &\text{ZoneCategoryName} = \text{“Room”} \wedge \\ &5.85 < \text{WallsalongPerimeter} \leq 19.45 \quad \Rightarrow \quad 9.56 < \text{ZonePerimeter} \leq 21.95 \end{aligned}$$

Other trials based on the contents of each room were performed so as to discover which objects normally reside together in similar classes of room. The association rules that resulted were no more interesting than those shown above. The mined rules have high confidence but that does not make them interesting. We would expect that if total window width in a room is zero then there would be zero window surface area. This illustrates a problem with association rule learning: we do not just require those rules with the highest confidence, we desire rules that are "interesting". However

association rule learners generally find all association rules and then return those with the highest confidence measure.

This is a known property of association rule learners (Bayardo & Agrawal 1999). Bayardo and Agrawal (1999) claim that statistical metrics such as support, confidence, entropy and χ^2 value are measures of "interestingness" and that using such measures can lead to efficient mining of rules. We believe that such measures may lead to statistically interesting rules, but that to acquire rules that are interesting to a domain expert requires something else. This something else is, we believe, based on domain knowledge, usually through preprocessing or through dumping lots of rules and then post-processing them. Indeed this is often done implicitly and without acknowledgement, with an attribute set and algorithm being tailored to suit a specific goal prior to running the learning algorithm.

3.3 SUPERVISED CLASSIFICATION

A number of classifiers were induced on the CAD data set. For the class label, the briefing database queries were used to transform the room name on each instance into one of the following: surgery, recovery, utilities, services, bathroom, corridor, consulting, reception, and other. Classifiers were then induced using 10-fold cross-validation.

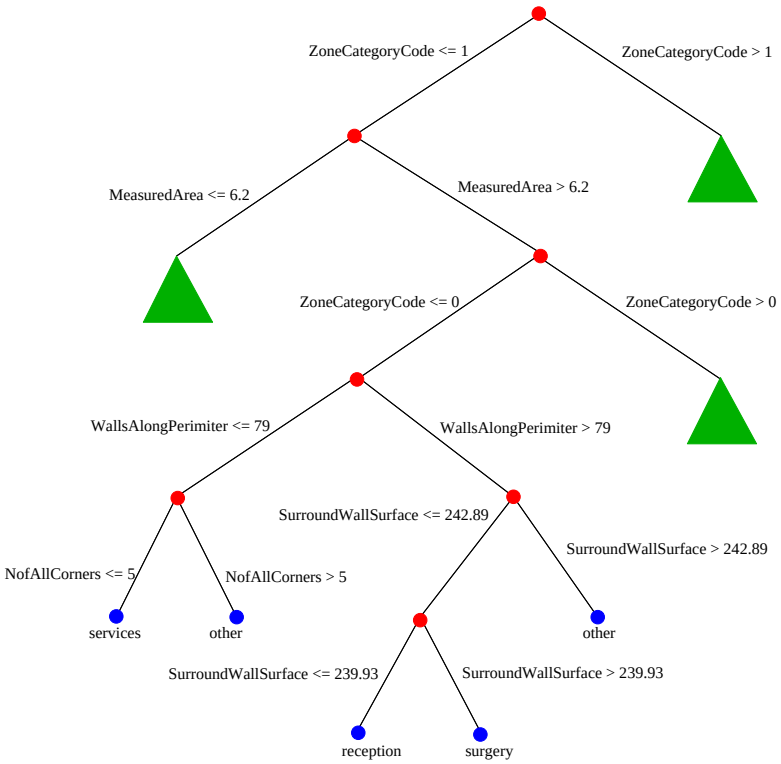

Figure 3. Induced decision tree fro hospital data. Part of the tree is shown explicitly as nodes and leaves, the remainder as compressed green triangles. Red nodes are decisions, blue are class labels.

As an example, Figure 3 shows part of a probabilistic decision tree induced using Quinlan's C4.5 algorithm. This tree was generated from inside the meta-classifier AdaBoostM1, with C4.5 pruning branches of confidence less than 25%. The boosting meta-classifier runs the tree induction algorithm

over a number of trials and a weighted voting scheme is used to assemble the final classifier. This tree contains 43 leaf nodes for the 9 classes, and it illustrates one of the common problems with decision tree induction: fragmentation. Indeed, running C4.5 alone and without pruning results in a tree with 80 leaf nodes and a lower classification accuracy. Cross-validation statistics showed that 93.9% of instances were correctly classified and classifications were shown to be strongly diagonalised when displayed as a confusion table. The results also reinforce the earlier comments about the nature of our data. A single architectural design may not contain enough instances to support full generalization, and sampling across designs may not be supportable.

As another example, consider a model tree that was constructed. Instead of inducing a symbolic classifier that maps from the attribute space onto an enumerated set of class labels, we learned the predictor of a numeric attribute. For illustration purposes we used the model learning algorithm M5' to learn a predictor of the numeric attribute MeasuredArea. The result is a small decision tree (the "tree" part, as shown in Figure 4) with an equation at each leaf with which to calculate the value of the attribute (the "model" part).

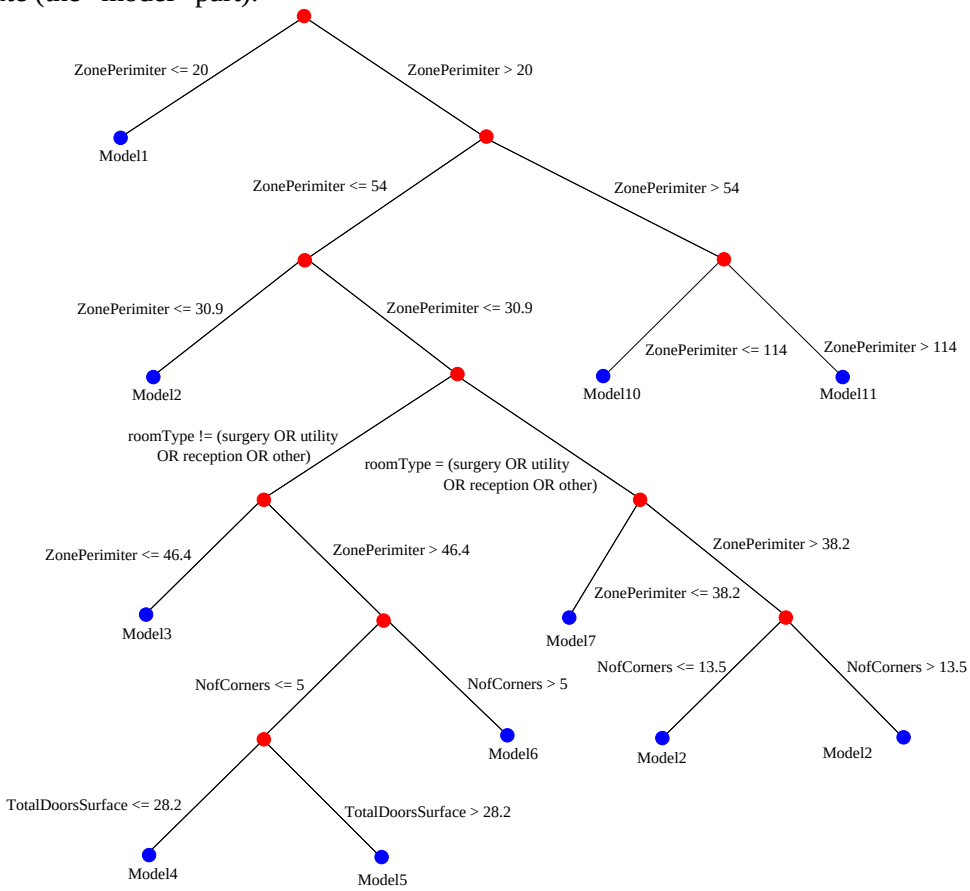

Figure 4. Model tree from hospital data

To use the classifier we proceed as for a decision tree until we reach a leaf. We then apply the equation referenced by that node. Thus the equation which calculates the target attribute MeasuredArea for instances with ZonePerimeter ≤ 20 is the equation labelled Model1, which is

```
MeasuredArea        =        6.65
```

```
                                - 0.318 * R1
                                + 0.442 * R2
                                - 0.607 * R3
                                + 0.0522 * ZonePerimeter
                                + 0.0033 * SurroundWallSurface
                                - 0.0356 * TotalDoorsSurface
                                + 0.00373 * TotalWindowsSurface
                                + 0.0832 * ZoneCategoryCode
                                - 0.0111 * TotalWindowsWidth
                                - 0.0532 * NofAllCorners

R1 = 1          ⇐               roomType=corridor ∨
                                roomType=other ∨
                                roomType=surgery ∨
                                roomType=utility ∨
                                roomType=reception
R1 = 0      Otherwise

R2 = 1          ⇐               roomType=other ∨
                                roomType=surgery ∨
                                roomType=utility ∨
                                roomType=reception
R2 = 0      Otherwise

R3 = 1          ⇐               roomType=surgery ∨
                                roomType=utility ∨
                                roomType=reception
R3 = 0      Otherwise
```

For most of the classifiers generated the classification accuracy was acceptable if appropriate induction parameters and meta-classifiers such as AdaBoostM1 were used. Indeed, a 100% accuracy may be unacceptable in some circumstances as it may indicate that it is not generalizing. The size of the C4.5 decision tree that resulted above is suitable for computer use as a classifier but a tree with 43 leaf nodes is not easily displayable to or understandable by a user. This problem of visualization is another known problem in data mining tasks (Hofmann, Siebes & Wilhelm 2000, Han & Cercone 2000). If the goal of an induction task is a machine learning one then provided that the classifier is efficient and accurate enough to be used on a target platform, all is well. However, if the task is knowledge discovery then the results should be understandable, both in content and in scope of results, by appropriately knowledgeable humans. Furthermore, as with the association rules, it is not clear how interesting such classifiers would be to a user.

4. Employing a conceptual model to facilitate knowledge discovery in design data

One of the primary goals of knowledge discovery is to convert raw data into “knowledge”. The number of abstractions on a nontrivial set of observations is unlimited. Which abstractions are chosen is a result of learning biases: limiting the representational language to those terms of interest, limiting

the order in which alternatives are examined, and so on. Indeed, as generalizing from a set of observations is not deductively valid (Langley 1996), without learning biases a system would have no reason to choose one abstraction over another and learning would not be possible. Thus, for data mining to work must require that some domain knowledge be applied, even if this is not recognised as such explicitly.

Pazzani (2000) describes knowledge discovery as a process of interaction between a data miner and a domain expert. Feedback on the novelty, utility and understandability of mined solutions lead to adjustments in data format and algorithm. For the techniques illustrated to be useful we should consider using domain knowledge. Domain knowledge can be used in the following ways:

Preprocessing stage

This is either unsupervised or supervised. Unsupervised includes techniques like feature selection, feature construction, discretisation, and clustering using some metric. Supervised involves either model based techniques (explicitly using domain knowledge), or providing the classes upon which discretisation and so on work. Further, decisions like whether to sample a database or not, and how to structure the data could be influenced by knowledge of the application domain.

A priori theory in data mining

Instead of starting with preprocessed data and then applying selected algorithms to uncover an underlying theory, we start with some kind of a priori theory and use the database to refine and extend it.

Inductive bias

This is either a representation bias or a search bias. As an example of a representation bias is effecting the choice of algorithm or representation based on domain knowledge. Examples of search bias are effecting instance selection order, encoding templates that constrain a search, and using user feedback during search. For this work the primary driver of inductive bias should be the underlying goals of the knowledge discovery task in this project.

Considerations about the representation of discoveries are also extremely important if we need to incorporate discovered knowledge back into a design support system. Our proposal is to use a conceptual model, a predefined knowledge representation schema, capable of accommodating different discoveries and presenting them back to the designers in an integrated form. Simoff and Maher (1998b) proposed a semi-automatic procedure for deriving terminological taxonomies by text mining of the case data in a hyper-media design case library. The conceptual model, around which the cases in the library were structured, provided the background knowledge in this method. The same structure can be used to accommodate extracted knowledge. Initially, Simoff and Maher (1998a) proposed a knowledge representation schema, ontology in this case, for multimedia data mining for design information retrieval.

If we take discovery to be to “divulge, reveal, disclose ... knowledge'” (The Oxford English Dictionary 1989) and take knowledge to be justified belief, then we might validly consider some methods which do not attempt to generalize over a data set to be knowledge discovery. For example, re-presenting CAD data in a form that reveals relationships to the user that would not otherwise be

visible facilitates knowledge discovery on the part of the user without such applications themselves performing any inductive or statistical generalizations explicitly.

In this work, the formalism that is used is based on the Activity/Space conceptual model, proposed by Maher, Simoff and Mitchell (1997) and further elaborated in a design ontology by Simoff and Maher (1998). Activity/Space ontology makes explicit the representation of the required functionality and resulting spatial arrangement and to some extent supporting structure in an architectural design. Building design knowledge, then, is either knowledge of activity (required functionality) or spatial knowledge (realized structure).

Figure 5 illustrates the structure of an activity/space model. Above the thick horizontal line is an activity hierarchy; below the line is a corresponding space hierarchy. An activity is "a purposeful action, whose performance requires a particular amount of space, time and an object that performs this activity" (Simoff & Maher 1998a). For our data set the high level activities are called HPUs and their requirements come from the briefing database. Examples include emergency care, surgery, medical ward, maternity and catering. Further, a medical ward requires the facilitation of lower level activities such as providing for a patient bedroom, waiting by visitors, storing of linen and equipment, and so on. A space acting as a bedroom then requires an appropriate spatial layout and set of objects for sleeping, laying, reading, eating, getting out of bed, and so on. Thus the requirements for a building can be expressed as an activity hierarchy. Similarly, a spatial model of a designed building can be expressed hierarchically based on topology, or on a classification of the activities supported, or both.

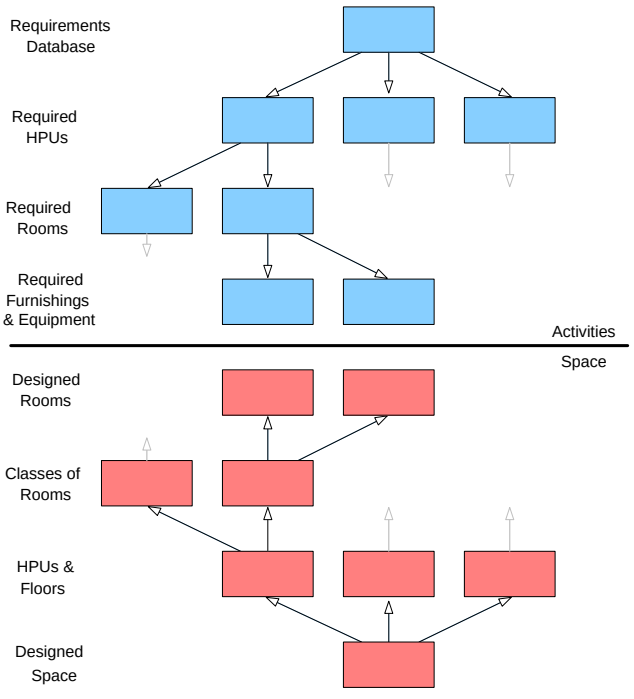

Figure 5: Example activity/space model of a hospital design. The light grey arrows indicate sub-hierarchies that are not drawn.

Now in our data set the room numbers used in the briefing database map directly to zone numbers in the CAD design. Suppose that we take the set of rooms in a briefing database (the design

requirements) and with domain expert help we derive a set of low level activities α_n. For each α_n, if we label each room from a sample training set of hospitals with the support or otherwise of activity α_n then we can learn a classifier for each activity. If we apply this set of classifiers to a particular hospital CAD test data set then we can classify the design accordingly and thereby derive a spatial hierarchy which corresponds to the activity hierarchy.

An example may make this clearer. Taking the data set and an arbitrarily chosen activity (in this case it is “acting as a bedroom”), a decision tree is induced using the method described. Each instance in the training set consisted of the possible contents of a room plus its nominal area. The resulting decision tree has 14 leaves, 21 nodes and an accuracy of 99% cross validated on the training set. Labelling rooms that support “acting as a bedroom” as class A and others as class B, we derive a spatial hierarchy as follows. The root is the entire space: the hospital. This is partitioned into floors, each floor is partitioned into HPUs, each HPU is partitioned into class A or B, and each class in each HPU is partitioned into rooms. We have, then, the CAD design re-represented as a functional hierarchy. Figure 6 shows part of the ambulatory care sub-hierarchy, visualized as an SGF graph. SGF is a Structured Graph Format presented to the viewer from an XML file (Liechti, Sifer & Ichikawa 1998, Sifer 1999). This XML specifies a set of nodes and two kinds of directed edges. In our case the nodes correspond to the nodes in the spatial hierarchy. For example, the following XML extract shows two nodes with display names “COFFS HARBOUR HOSPITAL” and “FLOOR NUMBER 1”.

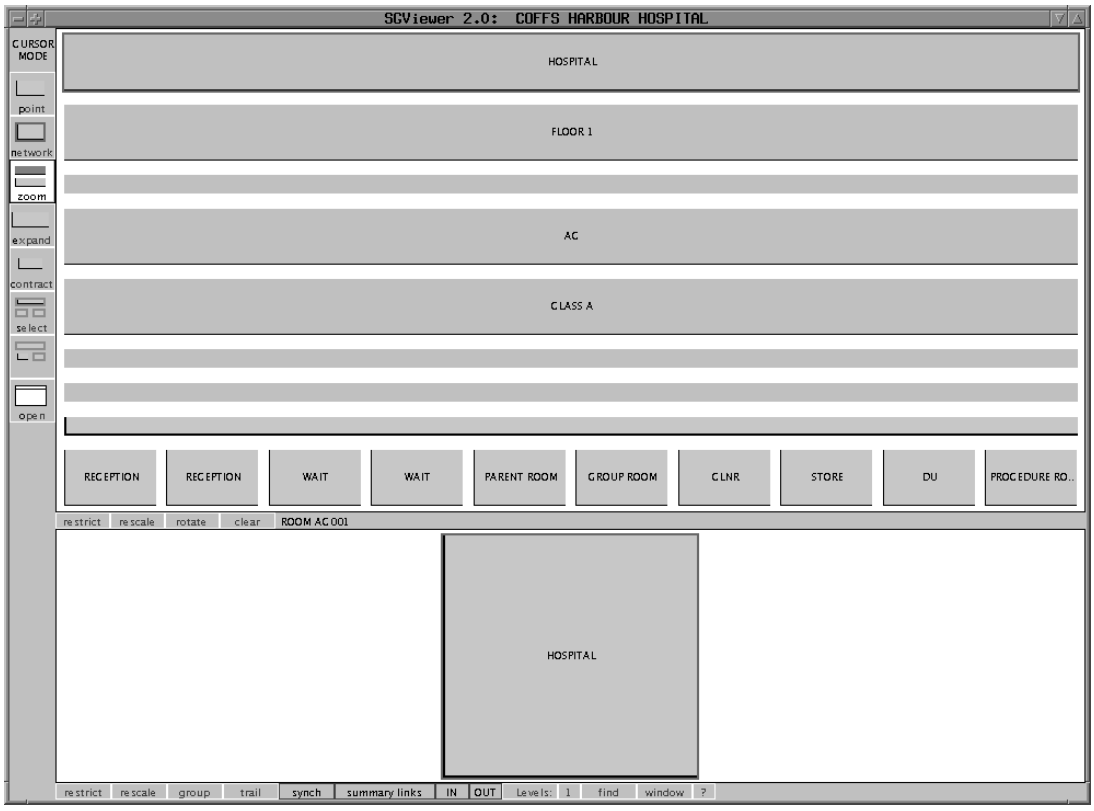

Figure 6: Part of the spatial heirarchy displayed as an SGF graph. The display is of some of the "ambulatory care" HPU rooms, zoomed into the room level.

<NODES>

```
<NODE NODEID="HOSPITAL" LABEL="HOSPITAL"></NODE>
<SGATT NAME="DESCRIPTOR" VALUE="COFFS HARBOUR HOSPITAL"/>
</NODE>
<NODE NODEID="FLOOR1" LABEL="FLOOR 1">
<SGATT NAME="DESCRIPTOR" VALUE="FLOOR NUMBER 1"/>
</NODE>
...
</NODES>
```

The first kind of edge link is hierarchical and thus describes a tree. These will specify edge links from the root node to each floor node, from each floor node to each HPU node, and so on. Figure 6 shows part of the hierarchy, with the SGF viewer set to zoom in on CLASS A rooms in the ambulatory care HPU on the first floor. The following extract shows the hierarchy edge links for the leftmost three rooms.

```
<HIERARCHY>
...
<LINK SOURCE="FLOOR1ACCLASSA" DEST="AC001"></LINK>
<LINK SOURCE="FLOOR1ACCLASSA" DEST="AC002"></LINK>
<LINK SOURCE="FLOOR1ACCLASSA" DEST="AC003"></LINK>
...
</HIERARCHY>
```

The second kind of edge link is associative and describes a directed graph. The ability of the SGF Viewer to simultaneously display and navigate both a hierarchy and a topology are one reason for choosing this tool. The kinds of association that could be displayed vary with what associations are of interest to the user. One example would be to display topology using the associative links and classification using the hierarchical links. Figure 7 shows as an example setting topology amongst HPUs and the associative links.

The following XML extract shows three associative edge links. If we look at the lower half of Figure 7 we can see node “HPUAC” in the centre in pink, input associations to “HPUAC” on the left in blue, and output associations on the right in blue. Thus the link from “HPUON” to “HPUAC” is visualized as a blue box labelled “ON” located the immediate left of a pink box labelled “AC”.

```
<NETWORK>
...
<LINK SOURCE="HPUON" DEST="HPUAC"></LINK>
<LINK SOURCE="HPUON" DEST="HPUPR"></LINK>
<LINK SOURCE="HPUHT" DEST="HPUAC"></LINK>
...
</NETWORK>
```

This XML file was generated from the ArchiCAD data using the following processes. Firstly ArchiCAD was used to dump a zone listing. This resulted in a table containing, amongst other attributes, the zone name and number, and zone contents. This was then processed into a table containing floor number, HPU code, room number and room name. The classifier was then run and an

extra column is appended to the table containing the results of the classification. This table then provides enough information to generate the node and hierarchy tags in XML format. For the associative network tags we chose, for this example, to use topology. If we extracted from the CAD data the pair of rooms that each door connected then we would have a set of associations describing room topology. For the example shown in Figure 7 an HPU A is marked as being associated with another HPU B if the distance between the centres of A and B is within 150% of the distance from the centre of A to the centre of its closest neighbour.

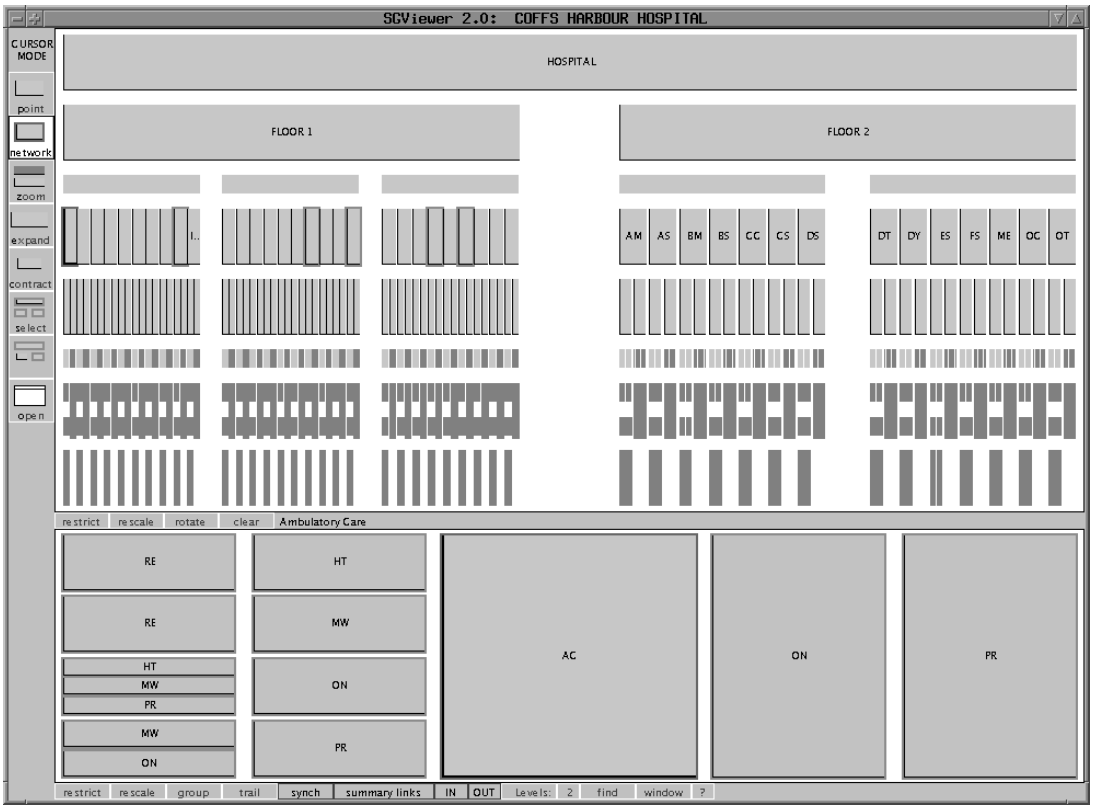

Figure 7: SGF Heirarchy showing the root node, and associations to and from the “AC” node of floor 1.

In summary, then, the process operates in two phases and resembles that shown in Figure 8. The first phase starts with test data for requirements and some domain knowledge. Here an activity model is acquired and applied, resulting in a table of test data with one attribute appended for each activity label. Machine learning techniques are then applied to the CAD data to learn a classifier for each activity. The second phase involves the application of the classifier to the selected test data, and a source XML data file is generated. A GUI allows for the selection of activities of interest by a user, and the source XML is transformed into an SGF format for display. This then iterates as user selections change.

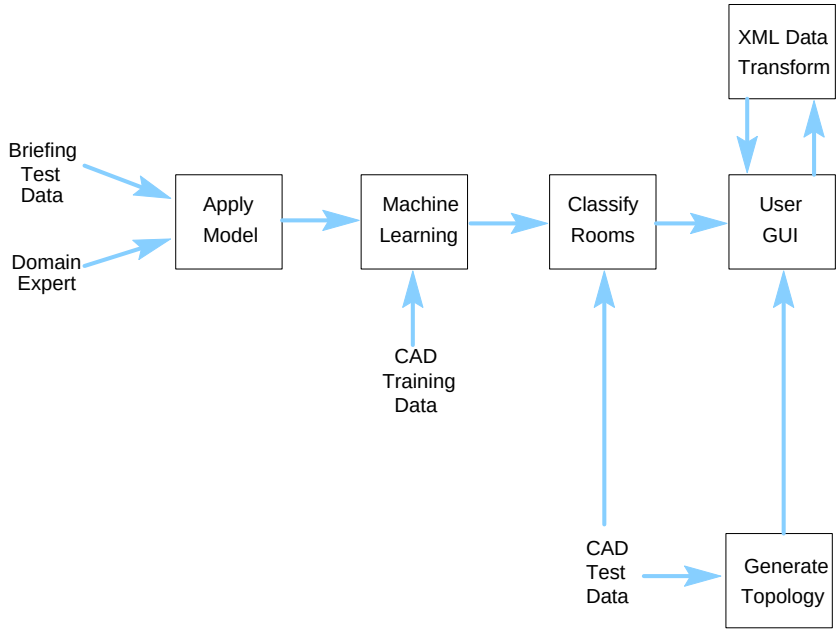

Figure 8: Summary of the Knowledge Discovery Process

5. Conclusion

We believe that there probably are underlying patterns in the architectural structure of similarly styled hospitals. However it is not clear yet how those patterns can be discovered and then presented in such a form that the "knowledge" would be interesting to practicing architects. Finding an underlying pattern to be statistically significant does not necessarily make it useful. Perhaps what is needed is further research into qualitative representations of architectural structure and style.

As to finding patterns in the contents of buildings, it is debatable whether there is any *real* pattern in the contents of similarly styled rooms and buildings. The presence of statistically significant patterns within a given data set does not necessarily generalize to other buildings of the same style or use. The patterns could also be due to an architect's preference, or just serendipity.

It is common for data mining papers to start with a data set already preprocessed into a set of useful attributes, and for a selected algorithm to then be run. If the data mining task is to take raw data and uncover underlying patterns then half of the work has been done without acknowledgement of the importance of the domain knowledge which was applied. The role of domain knowledge and learning biases should be acknowledged in any knowledge discovery work.

Perhaps a more immediately profitable use for data mining techniques on architectural CAD data is the re-presentation of designs in a different form according to an underlying ontology. We considered here the Activity/Space Ontology but others could be applied to those engineering domains that also design with CAD tools. Such a re-presentation then enables knowledge discovery *by the users* of the data. We believe that more work is needed on re-presenting and visualizing design knowledge in forms that are useful to designers.

Acknowledgements

This work has been supported by a grant from the Australian Research Council. The research has been conducted in collaboration with the architectural design company Woods Baggot – Sydney, who provided real world data and additional domain knowledge.

References

Bayardo, R. J. & Agrawal, R. (1999). Mining the most interesting rules, *Proceedings of the fifth ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 145-154.

Blumer, A., Ehrenfeucht, A., Haussler, D. and Warmuth, M. (1989). Learnability and Vapnik-Chervonenkis dimension. Journal of the ACM, 36(4), pp929-965.

Fayyad, U. M., Piatetsky-Shapiro, G. & Smyth, P. (1996). From data mining to knowledge discovery: An overview, in U. M. Fayad, G. Piatetsky-Shapiro, P. Smyth & R. Uthurusamy (eds), *Advances in Knowledge Discovery and Data Mining*, AAAI Press, Menlo Park CA, pp. 1-34.

Frank, E., Trigg, L. & Hall, M. (2000). WEKA. Java Programs for Machine Learning.

*http://cs.waikato.ac.nz/ ml

Han, J. & Cercone, N. (2000). RuleViz: a model for visualizing knowledge discovery process, *Proceedings of the sixth ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 244-253.

Hanney, K. and Keane, M.T. (1996). Learning adaptation rules from a case-base. In I. Smith, B. Faltings (eds), *Advances in Case-Based Reasoning*, Springer, Heidelberg, pp.179-192.

Haussler, D. (1988). Quantifying inductive bias: AI learning algorithms and Valiant's learning framework, *Artificial Intelligence*:36, pp 177-222.

Hofmann, H., Siebes, A. P. J. M. & Wilhelm, A. F. X. (2000). Visualizing association rules with interactive mosaic plots, *Proceedings of the sixth ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 227-235.

Langley, P. (1996). *Elements of Machine Learning*, Morgan Kaufmann, San Fransisco.

Liechti, O., Sifer, M. J. & Ichikawa, T. (1998). Structured graph format: XML metadata for describing web site structure, *Computer Networks and ISDN Systems* 30: 11-21.

Maher, M.L. (1997). SAM: A multimedia case library of structural designs. In Y.T.Liu, J-Y. Tsou and J-H. Hou (eds), *CAADRIA'97*, Hu's Publisher Inc., Taiwan, pp. 5-14.

Maher, M.L., Balachandran, B., Zhang, D.M. (1995). *Case-Based Reasoning in Design*, Lawrence Erlbaum Associates, New Jersey, 1995.

Maher, M. L., Simoff, S. J. & Mitchell, J. (1997). Formalising building requirements using an activity/space model, *Automation in Construction* 6: 77-95.

Maher, M.L. and Simoff, S. (1998) Knowledge discovery from multimedia case libraries, in I. Smith (ed) *Artificial Intelligence in Structural Engineering*, Springer, Berlin, pp 197-213

Nagy, G. (2000). Twenty years of document image analysis in PAMI, *IEEE Transactions on Pattern Analysis and machine Intelligence* 22(1): 38-62.

Pazzani, M. J. (2000). Knowledge discovery from data, *IEEE Intelligent Systems* 15(2).

Shavlik, J. W. and Dietterich, T. G. (1990). *Readings in machine learning*, Morgan Kaufmann, San Mateo, CA.

Sifer, M. (1999). Client side SGF Viewer.

*http://www.isl.hiroshima-u.ac.jp/projects/SGF/sgviewer.htm

Simoff, S. J. & Maher, M. L. (1998a). Designing with the activity/space ontology, in J. S. Gero & F. Sudweeks (eds), *Artificial Intelligence in Design '98*, pp. 23-43.

Simoff, S. J. & Maher, M. L. (2000). Analysing participation in collaborative design environments, *Design Studies* 21: 119-144.

Simoff, S. and Maher, M.L. (1998a) Ontology-based multimedia data mining for design information retrieval, Proceedings of the ACSE Computing Congress, Cambridge, MA.

Simoff, S. & Maher, M. L. (1998b). Deriving ontology from design cases. *Proceedings of Design Computing on the Net 98.*

*http://www.arch.usyd.edu.au/kcdc/journal

Simoff, S.J. and Maher, M. L. (1999). Knowledge discovery in hypermedia case libraries - A methodological framework, in G. Beydoun, P. Compton, A. Hoffmann and D. Richards (eds), *Proceedings of the Fourth Australian Knowledge*

Acquisition Workshop AKAW'99, in conjunction with 12th Australian Joint Conference on Artificial Intelligence, AI'99, Sydney, Australia, pp. **

Soibelman, L. & Kim, H. (2000). Generating construction knowledge with knowledge discovery in databases, in R. Fruchter, F. P. na Mora & W. M. K. Roddis (eds), *Computing in Civil and Building Engineering: Proceedings of the Eigth International Conference (ICCBE-VIII)*, pp. 906-913.

The Apache Software Foundation (2000). Xalan-java. The Apache XML Project.

*http://xml.apache.org/xalan-j/

The Oxford English Dictionary (1989).

Von Glasersfeld, E. *Radical constructivism : a way of knowing and learning*, Falmer Press, Washington, D.C., 1995.