Development of a Knowledge Acquisition Tool for Bridge Design ES

Amin Hammad\textsuperscript{+}, Yoshito ITOH\textsuperscript{+}

\textsuperscript{+}Nagoya University

Knowledge acquisition for developing design knowledge bases is difficult and time consuming. In this paper, a new approach for acquiring design knowledge is presented. The object-oriented approach is applied to model the design objects and the design process. It is discussed that in the case of routine bridge design, synthesis of the bridge structure can be done through recursive top-down refinement using selection among predefined types, and decomposition of the structure into predefined components. Having this design model, a methodology for using analogy to augment the knowledge base is proposed. Analogy can be used to retrieve and modify an available design object in order to create a similar object interactively. A prototype system that implements this methodology is presented.

1. INTRODUCTION

Expert systems (ES) technology has great potential in design problems. However, the process of knowledge acquisition for building ES is usually difficult, expensive and time-consuming. Many knowledge acquisition tools have been developed to allow the domain expert to converse directly with the ES [2]. Analogical reasoning has been applied to design problems [1, 3]. In this paper, we discuss the requirements of design expert systems and the representation of the design objects and design processes. We discuss that having a good design model can help in creating a library of design cases. When the designer needs to define a new design case, he can start by retrieving the most similar case from the knowledge base. The transformation from the analogical case to the new case can be done interactively. A prototype system called the Design Object Editor is under development as a research tool to test the previous ideas.

2. USE OF ANALOGY IN BRIDGE DESIGN

Designers tend to be comfortable using analogy when attacking new design problems. Even for routine design problems, analogy serves in clarifying similarities and differences among related design problems.

For instance, orthotropic plate girder bridges (hereafter box-type) and I plate girder bridges (hereafter I-type) share many similarities: (a) Structural similarities such as they have the same main structural components and they have the same possible variations. (b) Behavioral similarities regarding the loads they carry and the way they transfer these loads to the substructure. Typical cross sections of the two types are shown in Fig.1. The differences between the two types can also be classified in three categories: (a) Structural differences: the I-type and the box-type have different components. Figure 2 shows the components tree of the two types. (b) Functional differences: the box-type is usually used for longer span lengths. (c) Behavioral differences: the main behavioral difference between the I-type and box-type is that the box shape can resist torsion.

These similarities and differences can facilitate the development, maintenance and usage of design ES. This idea is very obvious for humans who use analogy as one of their basic learning strategies. We
can expect also that having design systems that can recognize the similarity between previously designed object and a new one, yet to be designed, will improve the performance of these systems.

3. REQUIREMENTS OF THE DESIGN SHELL

Knowledge acquisition has been recognized as a key problem in the development of ES. In particular, design problems are inherently ill-defined. The development of a design ES is typically done by a person(s) other than the expert designer because of the difficulty of using the specific ES tool. The following requirements are suggested to facilitate the development of design ES:

1. There is a need for a specialized design ES shell that can represent both the design objects and the design process in an intuitive and clear way.

2. The designer should be able to feed the knowledge base of the ES with the design knowledge directly, reducing the need for the knowledge engineer. To make this possible, the design ES shell should fulfill the following requirements: (a) User-friendliness: The man-machine interaction should consider not only that the expert may not know much about ES, but also that he may not know how to structure his knowledge about the domain of his expertise. (b) The ability of incremental and systematic growth of the knowledge base while keeping the integrity of the knowledge. (c) Flexibility in the development: The normal approach in design is the top-down approach. However, programming languages or ES tools may in many cases encourage or enforce a bottom-up strategy. (d) Other general management capabilities like version management and security checking.

4. THE DESIGN OBJECT EDITOR

4.1. System outline

The Design Object Editor is under development on a SUN workstation using Common-Lisp language. The system has two sub-modules: (a) The knowledge base management module: written using a frame-based and rule-based ES tool [4]. (b) The user interface module: developed within the X window system environment.

The system allows the design expert to add his knowledge to the KB easily, incrementally and consistently. This has been done by following the requirements mentioned in section 3. The system has three main components: (1) class editor, (2) rule editor and (3) instance editor. The user can call the different functions of these components through pull-down menus. The user input is checked by the Input Knowledge Checker. The class editor allows editing classes, attributes, bitmap images and constraints.

4.2. The user interface

Figure 3 is a hard copy of the screen showing the main window of the system. The main window has a number of scrollable menus, pull-down menus and state fields. These elements are described briefly in the following.

1. Scrollable menus: (a) The Types menu – displays a list of the names of all physical objects of the selected knowledge base indented
to reflect the parent-child relationship. (b) The Parts menu – displays the same list of objects’ names, but from the point of view of whole-part relationship. (c) The Attributes menu – displays the list of the attributes of the selected object. (d) The Cardinality menu – displays the list of the possible values of the selected attribute. (e) The Value menu – shows the actual value of the selected attribute. (f) The Instances menu – displays the instances of the selected class.

2. Pull-down menus: General menu has main functions to create, load or save a knowledge base. Select, Add, Delete, Change and Copy menus are used to edit the design objects and there attributes. The Rule menu has sub menus that facilitate the creation of forward-chaining and backward-chaining rules interactively.

3. State fields: The state fields shows the current selections made by the user such as the current knowledge base, class, attribute and instance. The unit and default value of the selected attribute are also shown.

4.3. The design model

4.3.1. Design object model

Each physical object model corresponds to a class. A class may be a sub-class of one or more physical or abstract objects. This relation between a sub-class and its super-class is called "is-a" or "kind-of" relation. For each non-leaf class, we have special attributes that are called classifiers. A classifier is an attribute that, according to its value, instances of the class can be further classified into new subclasses. For example, the bridge class may have many classifiers: Section_Type, Deck_Material, Structural-type, etc. as shown in Fig.4. For each classifier, a number of sub-classes may be generated automatically. For instance, most of the Section_Type, Structural_Type and Deck_Material possible values may be combined to create useful new classes like Concrete_Deck_Bridge, Concrete_Deck_Bridge, etc. This action of generating new classes is called expanding a class according to a classifier.

A physical object model may have any number of components, and may be a member of another object. This relation is called "part-of". The relations between objects in Fig.2 are good examples of part-of. This decomposition is possible through the use of another special attribute called decomposer. The decomposer value is a list of all components of the class. It is not available for the basic components of the object which are not supposed to be divided into smaller parts.

4.3.2. Design process model

The design process is usually modeled using one or more plans to represent the different operators that should be applied on the entities to transform, decompose and refine them in a way that insures that the final product will satisfy the initial design specifications. A plan can be decomposed further into tasks and steps to reduce its complexity in parallel to the design object decomposition.

The major design tasks in bridge routine design are (1) synthesis of the structure, (2) structural analysis and (3) evaluation of the design. The knowledge used in the design process varies greatly depending on the design task and the design phase.
For instance, in preliminary design, synthesis and evaluation are based on heuristic rules and only rough analysis is made. However, in detailed design, synthesis is merely selection among known alternatives and analysis is done according to precise algorithms. Evaluating the detailed design is done by comparing the simulated behavior of the structure with the requirements of the design specifications.

The previous knowledge needed in the bridge ES design can be represented in the Design Object Editor as follows:

1. Bridge structure, function and behavior: They are represented by classes and attributes. Classifiers and decomposers are used during the design to refine the design object.

2. Design specifications knowledge: They are constraints represented by decision tables and embedded within the design object classes. The decision tables can be edited interactively.

3. Design plans: We use design plans to express the well understood part of the design process which can be organized as tasks and sub-tasks. These tasks are implemented by procedural subroutines coupled with the design object, i.e. for each component object, the design tasks and sub-tasks related to this object are described as methods of the object.

4. Heuristics design rules: These rules are grouped in rule sets and embedded within the design object classes.

4.4. Creating new classes by analogy

In this section, it is shown how the designer can use the system to define a box plate girder by copying and amending the object of an I plate girder. The user can browse through the menu of already defined classes to find a similar class which he can copy to the name of the new class and then edit it as needed. However, as the number of defined objects increases, this process becomes difficult. To facilitate retrieving the similar cases that can be used to make the analogical reasoning, the user can start by giving the name of the new class. This name will be analyzed and a number of keywords will be extracted from it. For instance, if the user enter the object name "Box_Plate_Girder", the word "Plate_Girder" will be recognized by the system as a keyword and the objects that contain this keyword will be retrieved. In this example, the following two objects were found: Plate_Girder and I_Plate_Girder. The user can easily select the later object as a similar case.

The copy function will result in making a duplicate of the L_Plate_Girder object with a new name: Box_Plate_Girder. The user can delete the components which are not needed (the vertical and horizontal stiffeners and the web) and add the new components (the left and right webs and the ribs). This will complete the geometrical modeling of the new object. The user can also change the definition of the tasks used in the design. The sub-tasks will change to reflect the changes in the structure. The details of each of these sub-tasks will be defined in the components classes. For instance, torsion was not considered in the case of the L_Plate_Girder object but it should be considered in the case of Box_Plate_Girder. The example given here is very simple and its target is to show that all the mentioned changes can be done interactively. Further details are implemented in the procedural subroutines which are called from within the system.

5. CONCLUSIONS

In this paper, the problem of knowledge acquisition for bridge design was discussed. It has been shown that in order to facilitate the development of bridge design ES, there is a need for a specialized design shell that is user-friendly, allows the incremental and systematic growth of the knowledge base and flexibility in the development. The object-oriented approach was used effectively to represent the design object model and the design process model. Classification and decomposition of the design objects were used and implemented in a prototype system. Using the previous model, analogical reasoning was suggested as a powerful method in augmenting the design knowledge base.

References


1992年度
人工知能学会全国大会（第6回）論文集
Proceedings of the 6th Annual Conference of JSAI, 1992

発行日 1992年6月23日
発行所 法人人工知能学会
東京都新宿区渋谷4-7 〒162
オフィスビル402 ☎ (03)5261-3401