Artificial Intelligence for Engineering Design, Analysis and Manufacturing (1998), *12*, 463–478. Printed in the USA.
Copyright © 1998 Cambridge University Press 0890-0604/98 $12.50

A design-plan-oriented methodology for applying case-based adaptation to engineering design

TALAL AL-SHIHABI AND IBRAHIM ZEID

Department of Mechanical, Industrial, and Manufacturing Engineering, Northeastern University, Boston, MA 02115, U.S.A.

(RECEIVED July 16, 1997; ACCEPTED July 2, 1998)

Abstract

Adaptation of design cases is usually the most challenging part in building any case-based reasoning design system. The success of the adaptation process in finding a solution for a new design problem determines the success of the entire case-based reasoning (CBR) system. The techniques used for generating design solutions have many common aspects among the various engineering design classes that make them amenable to be captured in a generic framework for an acceptable level of abstraction. This paper proposes a design-plan-oriented methodology for adapting design cases to produce a solution to a new design problem in the domain of engineering design. The proposed methodology uses multicase adaptation and case built-in adaptation knowledge to produce a design plan for a new design problem. We first define the model of case representation to work with the proposed methodology. We then define the overall structure of the procedural framework of this methodology and its subprocesses. The system is then demonstrated through an application from the structural engineering domain.

Keywords: Design Adaptation; Case-based Reasoning; Engineering Design; Case-based Adaption

1. INTRODUCTION

The solution of an engineering design problem is a design plan of an artifact that satisfies predefined goals and constraints. Engineers involved in solving a conceptual design problem rely heavily on their experience in solving similar past problems. According to the availability of any relevant past experience, engineering design is classified mainly into two groups: creative design versus routine design. Creative design is characterized by the lack of past experience, while routine design is characterized by the availability of past experience in both what to design and how to design. This characteristic of routine design holds the key benefits of using knowledge-based reasoning in engineering design.

The question of how to represent past experience and what techniques to be used to utilize this experience defines the border lines between the different knowledge-based reasoning methods. Case-based reasoning (CBR), among knowledge-based reasoning methods, suggests the use of past experience in the form of a collection of cases; each case contains a description of a problem and its solution. The major advantage of using CBR over using other knowledge-based reasoning methods is that CBR reasons, when faced with a similar problem, from an initial state that is closer to the goal state instead of trying to achieve the goal state from scratch using rules or any other kind of compiled knowledge.

CBR has been applied to a variety of problems in many fields. Some examples are: CYCLOP for landscape design (Navinchanndra, 1988), JULIA for meal design (Shinn, 1988; Hinrichs, 1989), CADSYN for structural design of buildings (Maher & Zhang, 1993), DEJAVU in the domain of mechanical design (Bardasz & Zeid, 1993), FIRST for structural redesign of beams (Daube & Hayes-Roth, 1990), and CASETOOL as a CBR framework in the domain of engineering design (Shiva Kumar & Krishnamoorthy, 1995).

Once a design knowledge base is available, a CBR system is expected to perform three main processes. These three processes, usually implemented in this order, are design case retrieval and best case selection, design case adaptation, and design case storage. The design case retrieval and best case selection process furnishes the design process with a set of design cases, retrieved from the knowledge base, similar to the new design problem at hand. The design case adaptation then uses these cases in producing a solution or a design

Reprint requests to: Talal Al-shihabi, Department of Mechanical, Industrial, and Manufacturing Engineering, Northeastern University, 360 Huntington Ave, Room 334, Boston, MA 02115, U.S.A.

https://doi.org/10.1017/S0890060498125052 Published online by Cambridge University Press

plan for the new design problem. Finally, the design case storage adds the new generated design plan along with the new problem specification to the knowledge base as a new design case.

A closer look at CBR processes shows the importance of the adaptation process. As an intermediate process, its input comes from the case retrieval process and its output goes to the case storage process; its success determines the success of the entire CBR system more than the success of any other process. However, adapting design cases is one of the most challenging tasks in building any CBR system. The two major questions in adaptation are how to find what needs to be adapted and then what strategy to use to adapt correctly. There is no direct answer for either one of these two questions in most, if not all, cases.

Design problem specification and design solution generation techniques have many common aspects in the various classes of engineering design. An engineering design problem can be defined in the following way. Given a set of design goals, a set of design constraints, and a set of design data, find a design plan that creates, or proposes how to create an engineering artifact, either a component or a system, that satisfies the goals and the constraints (Bardasz & Zeid, 1991). This shows that the problem definition and solution generation techniques in engineering design are amenable to be captured within a general framework at an acceptable level of abstraction.

This paper proposes an adaptation methodology that uses case built-in adaptation knowledge and multicase adaptation techniques to generate a design plan for a new design problem. This methodology addresses the issues of the adaptation process in the domain of engineering design in terms of knowledge and strategies. The strategies are integrated into a computational model that is implemented in ANSI C. After reviewing the adaptation process in CBR in Section 2, the case representation model utilized by the proposed methodology is defined in Section 3. The procedural framework of this methodology and its subprocesses are then explained in Sections 4 through 8. The methodology is then demonstrated through examples from the structural engineering domain in Section 9.

2. ADAPTATION IN CASE-BASED REASONING

Adaptation of old cases is one of the major responsibilities of any CBR system. Because there is no old problem that is completely identical to a new one, CBR takes the solution of the most similar retrieved case from a knowledge base as an initial state and then tries to achieve the goal state, a solution for the new problem, through iteratively adapting this initial state. The adaptation process is complex and usually involves many decisions that are all crucial to the success in achieving a solution for a new problem. The most important issues in the adaptation process are identifying inconsistent parts that need to be adapted and then choosing a strategy to adapt them.

2.1. Identifying inconsistent parts

One of the most important advantages of using CBR is that reasoning begins from an initial state that is supposed to be closer to the goal state than a null state. To maintain this advantage, a decision is needed at an early stage about what to keep and what to change in the initial state. The accuracy of this decision determines the amount of effort needed to reach the goal state. Many methods have been used in making this decision in the various existing CBR systems. These methods are listed and explained in Kolodner (1993).

One important method is identifying the parts of the initial state related to the design aspects that are different between the new problem and the old one as inconsistent parts. This requires that the case be represented in a way that shows the relationships between all parts of the solution, on one hand, and the problem specification, on the other hand. A typical example that uses this method is CASEY (Koton, 1988). CASEY uses the differences in symptoms between two patients to define what part of the old diagnosis needs to be fixed.

Another method is to identify the parts of the initial state that are undesired in the goal state as inconsistent parts. CLAVIER changes the layout configuration of plane parts by identifying the parts in the initial layout that are not required in the goal state defined by the new problem specification as inconsistent parts that need to be fixed (Barletta & Hennessy, 1989; Hennessy & Hinkle, 1992).

It is common for a CBR system to use more than one method in making a decision on what needs to be fixed. In this case, the CBR system may choose one method or more to apply according to the case under study.

The first method mentioned in this section, which is defining inconsistent parts according to the differences between the specifications of the new problem and the old one, is used in the proposed adaptation methodology. As will be shown in the case representation model in the next section, each procedure in the design plan has three lists that identify what input and output groups are involved in this procedure and as a result what design aspects are related or affected by this procedure. If one or more of these aspects has a different definition in the new problem from that in the old case, then any procedure related to this aspect or any subdesign plan that contains such a procedure is considered inconsistent with the new problem and should be fixed accordingly.

2.2. Choosing an adaptation strategy

After deciding on what in the initial state needs to be adapted, the next step is to choose an adaptation strategy. This choice depends on the different aspects of the adaptation problem, including the scope of the adaptation and whether it is on the structural or the analytical level; the case representation model; the form of adaptation whether it is adding, deleting, or updating, etc.

https://doi.org/10.1017/S0890060498125052 Published online by Cambridge University Press

Many adaptation strategies have been used successfully in the various implemented CBR systems. These strategies are grouped as following :

1. substitution methods;
2. transformation methods; and
3. special purpose adaptation and repair.

Substitution methods substitute values suitable for the new problem for the inconsistent values in the old case. Substitution can be done in many different ways. CHEF, for example, adapts solutions by substituting values within an explicit framework of an old recipe (Hammond, 1986). JUDGE substitutes new values for old values in accordance to the differences in parameters between the old case and new problem (Bain, 1989). CLAVIER uses case-based substitution where a new value, suggested by another case, substitutes an inconsistent value in the old solution (Barletta & Hennessy, 1989; Hennessy & Hinkle, 1992).

Transformation methods transform a solution, or part of it, to make it appropriate for the new problem. Some examples here are JULIA (Shinn, 1988; Hinrichs, 1989) and KRITIK (Goel & Chandrasekaran, 1989). Special purpose adaptation methods are domain-dependent methods that take advantage of the special characteristics within a domain. PANDA adapts a solution using parts of old solutions, domain knowledge, and case histories (Roderman & Tsatsoulis, 1993).

A successful approach usually needs to combine more than one strategy and to provide heuristics to choose the most suitable one to apply under certain conditions.

The case-based substitution method is used as the main adaptation strategy in the proposed methodology. When a part of the design plan is found inconsistent, the system looks for alternatives in a group of similar cases. If a better alternative is found, the system will substitute it for the inconsistent one. This process guarantees to minimize inconsistencies in the design plan but does not guarantee to remove them completely. For this reason, another adaptation method can be used after applying the first one. This method is a structural modification of the design plan by adding or deleting procedures or subdesign plans. After all, an approach toward constraint satisfaction is used to repair imperfections that appear during the evaluation of the design plan.

3. THE CASE REPRESENTATION MODEL

The main issues of design case representation in general are: the contents and organization of a design case, the overall organization of design cases in memory, and the interaction between design cases and the user or the designer (Maher, Balachandran, & Zhang, 1995). The issues of the overall organization of design cases in memory and the interaction between design cases and the user are not discussed here because they have no direct impact on the case adaptation process.

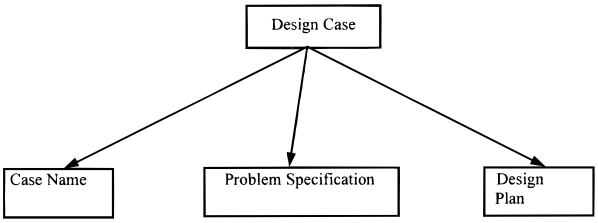


Fig. 1. The contents and organization of design cases.

3.1. The contents and organization of design cases

A design case should contain both the specification of a design problem and the solution of this problem. The contents of a design case are shown in Figure 1.

Problem specification is shown in Figure 2. This part describes the goals to be achieved, input and output attributes, and user-defined constraints on the goals and the values of design attributes, if there is any. Input and output attributes in this model are divided into groups, each group has the structure shown in Figure 3.

In general, a group should contain attributes that describe a design aspect within an engineering design class. For example, a group may be used to describe the behavior of an artifact associated with a design case, for example, a group named STRESSES contains attributes that define the types and values of stresses in the artifact. Another group may be used to define the geometric shape of this artifact by containing attributes that define the cross-section type, its dimensions, etc.

Input and output groups have the same structure, and input and output attributes are defined in the same way. However, values of output attributes are unknown until the problem is solved. In addition, some output attributes are

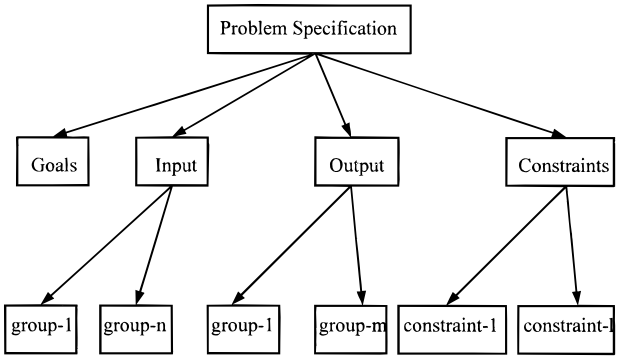


Fig. 2. The contents and organization of problem specification within a design case.

https://doi.org/10.1017/S0890060498125052 Published online by Cambridge University Press

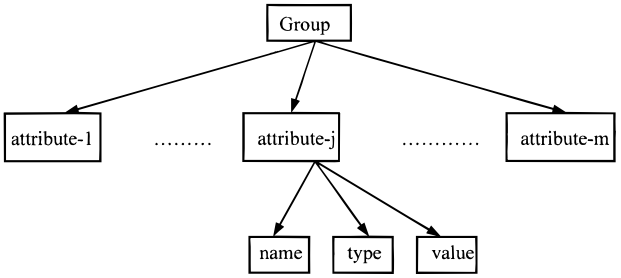


Fig. 3. Structure of input and output groups.

themselves unknown until a solution or a design plan is built for the problem.

Some of the attributes that describe one design aspect, and as a result should form a group, may be input attributes while the others are output attributes. As an example, the type and dimensions of a cross section describe one design aspect that is the cross section of an element. However, in some problems the cross-section type might be an input attribute while the dimensions of the cross section are output attributes. It is important in this case to divide these attributes into two groups: one as an input group and the other as an output group. This eliminates any confusion or unnecessary processing during the retrieval and the adaptation of design cases when a part of the solution of an old case is a part of the requirements of the new problem or vice versa.

Design constraints are the guarantor of the feasibility of the solution of a design problem. Within each engineering design class, a set of global constraints should be defined. In addition, the user or the designer has the possibility to add user-defined constraints that are specifically applied to a particular problem. Global and specific constraints in the proposed methodology have the structure shown in Figure 4. A logical relation may be any of: = for equal, > for greater, >= for greater or equal, < for less, <= for less or equal, and != for not equal. A logical operator may be any of: AND, OR, and NOT.

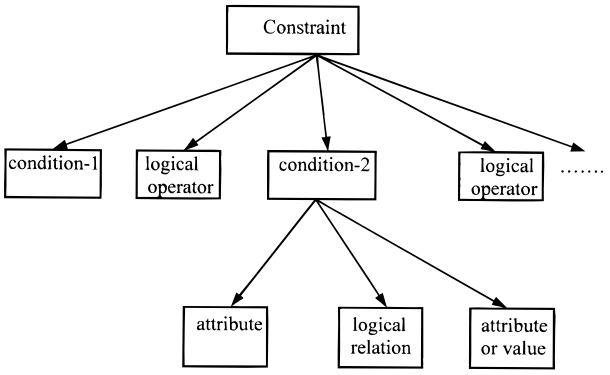


Fig. 4. Structure of design constraints.

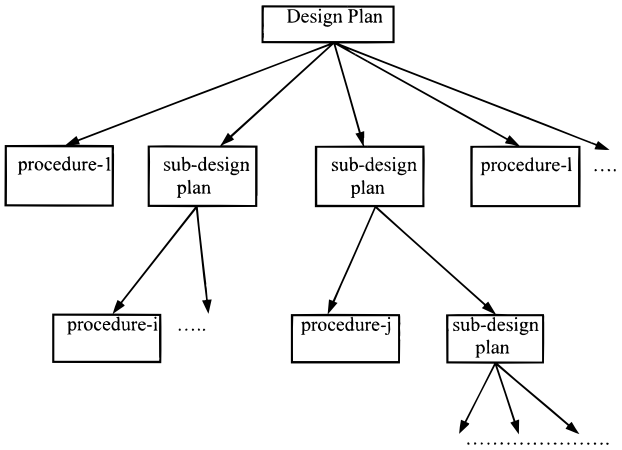


Fig. 5. Organization and contents of design plans within design cases.

The second major part of a design case is the solution of the problem defined in the first part. The solution here is defined as a design plan that can be replayed at any phase of the design process. The design plan consists of subdesign plans and procedures as shown in Figure 5. Each subdesign plan has the same structure as the main plan. Each procedure, Figure 6, has a list of procedure inputs that contains the input attributes used within the procedure body, a list of procedure intermediate inputs that contains the output attributes computed by other procedures and used as inputs within the procedure body, a list of procedure outputs that contains the output attributes computed by this procedure, and the procedure body that contains the instructions to be executed.

This structure of the design plan provides the adaptation module with information about the relationships between the solution parts themselves, and the relationships between any part of the solution, on one hand, and the problem specification, on the other hand.

3.2. An example

An example of the case representation model is a case that describes designing the simply-supported beam shown in Figure 7.

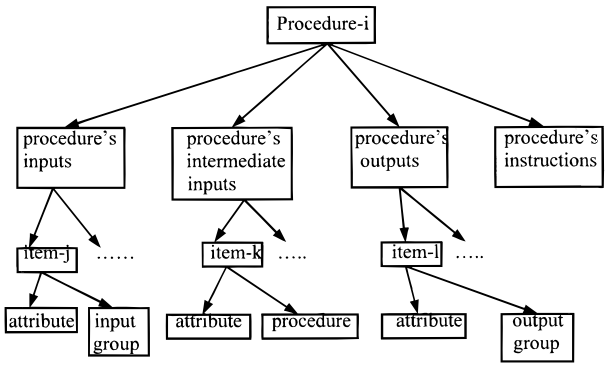


Fig. 6. The structure of procedures within design plans.

https://doi.org/10.1017/S0890060498125052 Published online by Cambridge University Press

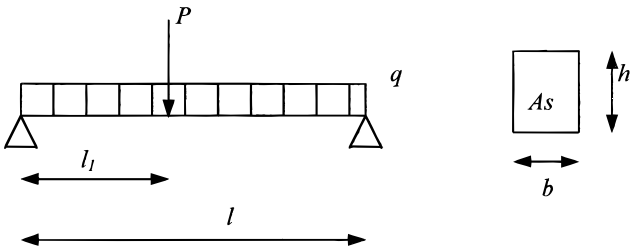


Fig. 7. Structural beam.

The inputs of the problem are grouped into five input groups. These groups are LOADS, SUPPORT, GEOMETRY, CROSS SECTION, and BUILDING MATERIAL. The output attributes are grouped into two groups, CROSS SECTION and STRESSES. The designer wants the total cross sectional area of tension steel not to exceed the value A_t. The case contents and organization are shown in Table 1.

4. A DESIGN-PLAN-ORIENTED METHODOLOGY FOR CASE-BASED ADAPTATION IN ENGINEERING DESIGN

The proposed design-plan-oriented methodology is based primarily on the case-based adaptation method. Case-based adaptation is an adaptation strategy classified among substitution methods. It suggests adapting an old solution by substituting the parts that are inconsistent with the new problem by alternatives from other cases.

The overall structure of the procedural framework of the adaptation methodology and the relationships among its parts are shown in Figure 8. The adaptation process in this framework is carried out by four subprocesses, the mapping pro-

Table 1. *Simply supported beam design case*

Case name:	BEAM-1		
Goals:	Resist bending moment by a simply supported beam		
Input groups:			
LOADS:	float	distributed-load	*q*
	float	concentrated-load	*p*
SUPPORT:	char	left-end	*"simply supported"*
	char	right-end	*"simply supported"*
GEOMETRY:	float	span-length	*l*
	float	concentrated-load-location	l_1
CROSS-SECTION:	int	cross-section-height	*h*
	int	cross-section-width	*b*
BUILDING-MATERIAL:	char	building-material-type	*"reinforced concrete"*
	float	concrete-characteristic-strength	*Fcu*
	float	steel-characteristic-strength	*Fy*
Output groups:			
CROSS-SECTION:	float	effective-height	*d*
	float	tension-reinforcement-area	*As*
	char	tension-reinforcement-selection	*"nTd"*
STRESSES:	float	max-design-bending-moment	*M*
	float	max-concrete-compression-stress	σ_c
	float	max-steel-tension-stress	σ_s
Constraints:			
CONSTRAINT-1	tension-steel-area $< A_t$		
Design plan:			
CALCULATING-DESIGN-MOMENT:			
Inputs:		distributed-load	group:LOADS
		concentrated-load	group:LOADS
		span-length	group:GEOMETRY
		concentrated-load-location	group:GEOMETRY
Intermediate inputs:		none	
Outputs:		max-design-bending-moment	group:STRESSES
Instructions:			
max-design-bending-moment = ...			
.....			
CALCULATING-ULTIMATE-MOMENT-OF-RESISTANCE:			
Inputs:	.....		
.....			
.....			

https://doi.org/10.1017/S0890060498125052 Published online by Cambridge University Press

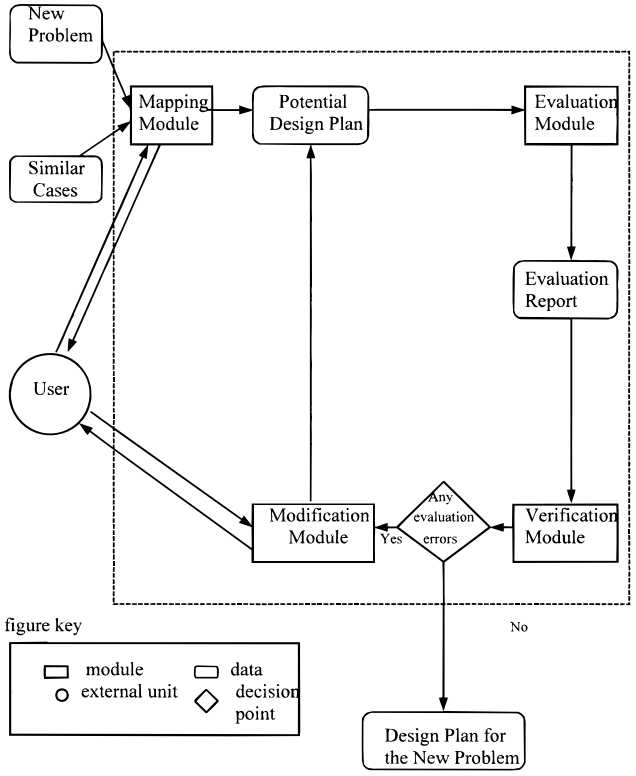


Fig. 8. The framework of the adaptation methodology.

cess, the evaluation process, the verification process, and the modification process.

The inputs to the framework are the new design problem specification and a set of similar cases that resulted from the case retrieval and best case selection process. The output of the framework is a solution, that is, design plan, for the new problem. The retrieved similar cases are ordered according to their degrees of similarities so that the most similar case is the first in the list.

The mapping process is the first to run within this framework. The solution of the most similar case is considered as a proposed solution for the new problem. By identifying the differences in specification between the new problem and the most similar case, the mapping process decides on what parts of the proposed solution need to be fixed. The solutions of the other similar cases are considered the main sources of alternatives to resolve inconsistencies between the new problem specification and the proposed solution. This process is complete when the proposed solution becomes a potential solution that is completely consistent with the new problem.

After a potential solution is generated, the evaluation process evaluates the solution for the assigned input values of the new problem. The purpose of the evaluation process is to compute the values of output attributes. In addition, the evaluation process catches any hidden errors within the potential design plan that could not be detected during the mapping process, such as using an attribute in the design plan in an inconsistent way with its defined type in the problem specification part.

The verification process then uses the values of the output attributes computed by the evaluation process to check if there is a violation in any relevant constraint either a library, global, constraint, or a user-defined constraint. All violations are reported to the modification process through a verification report.

The modification process is responsible of any postevaluation repair in the potential design plan. After the required repair is done, the new potential solution is forwarded again to the evaluation process to go through the loop again. When there is no postevaluation repair required, the adaptation process is considered complete and the resulted solution is considered a final output of the process.

The four subprocesses of adaptation are explained in detail in the next four sections.

5. THE MAPPING PROCESS

Mapping is the stage of adaptation that is responsible of producing a potential design plan for the new problem. Judging in this stage relies completely on the differences and the similarities between the new problem, on one hand, and the similar old cases, on the other hand. The solution of the most similar case is taken as the initial solution of the new problem. Inconsistent parts of this solution are then identified to be fixed or changed.

5.1. Identifying inconsistent parts

Each procedure in the design plan is checked to determine if it needs adaptation or not. This is done by defining the input groups involved in this procedure. Each input group in the new problem is compared with its correspondent one in the most similar case. If there is full match in all input groups involved in this procedure between the two problems, the procedure is considered acceptable for the moment without need for any change. However, if there is a difference in the contents of any group between the two problems, then this procedure and/or its correspondent subdesign plan need to be changed or fixed.

5.2. Adapting the inconsistent parts

Once a procedure in the proposed design plan is found to be inconsistent with the new problem specification, then this procedure needs to be changed or fixed. This is done by searching the other similar cases for a better alternative and map it to the proposed design plan. Additional postmapping repair is required in most cases.

5.2.1. Selecting the best alternative

The other similar cases are searched for a procedure that does the same thing as the one that needs to be fixed. Every

https://doi.org/10.1017/S0890060498125052 Published online by Cambridge University Press

time a procedure is found, it is listed as an alternative with its case assigned a number.

After all alternatives are listed, a comparison is done, for input groups involved in this procedure, between their correspondent cases and the new problem. If for any procedure there is a full match for all groups between the two problems, then this procedure substitutes the inconsistent one in the proposed design plan and the work will move directly to the postmapping repair. If there is no full match case for all the alternatives, then the best procedure becomes a question of the best case in terms of only the input groups involved in this procedure. In general, this case is not always the most similar case to the new problem as a whole. For example, case a and case b may be very similar as a whole but completely different in one part, case c may be unsimilar to both as a whole but completely similar to case a in terms of that part alone. The best case here can be defined as the case that has an alternative that can be repaired easily more than any other alternative can be.

To decide on this case we need to compare the input groups under question between the new problem and each case. For each input group involved as a criterion, by comparing the contents of this group in the new problem with its contents in another case, we get one of the following relations :

- *kind-1:* The contents of the group are identical in the new problem and the old case. This means that group definition in the new problem is the same as its definition in the old case.
- *kind-2:* Some attributes in the old case are not in the new problem, while all attributes in the new problem are in the old case. This means that this group definition in the new problem is highly recommended to be considered as a special case of its definition in the old case.
- *kind-3:* Some attributes in the new problem are not in the old case while all attributes in the old case are in the new problem. This means that this group definition in the old case is highly recommended to be considered as a special case of its definition in the new problem.
- *kind-4:* Some attributes in the new problem are not found in the old case and some attributes in the old case are not found in the new problem. This means there is no clear relation that can be inferred between the two definitions of the group.

After defining what kind of relation each involved input group has, the following heuristic rules are applied to choose the best case:

- *rule1:* Choose the cases whose respective procedures satisfy the pre-evaluation constraints within the class of engineering design to which the new problem belongs.
- *rule2:* If applying *rule1* resulted in choosing more than one case, choose the case that has the highest number of groups with relation of *kind-1*.
- *rule3:* If applying *rule1* and *rule2* resulted in choosing more than one case, choose the case that has the highest number of groups with relation of *kind-2*.
- *rule4:* If applying *rule1*, *rule2*, and *rule3* resulted in choosing more than one case, choose the case that has the highest number of groups with relation of *kind-3*.
- *rule5:* If applying *rule1*, *rule2*, *rule3*, and *rule4* resulted in choosing more than one case, choose any case randomly.

Before the alternative in the chosen case is mapped to the proposed design plan, its list of intermediate inputs is checked. This list contains the outputs attributes required as inputs by this procedure with the names of the procedures in which these attributes were computed. If any of these procedures was not already in the proposed design plan, then this procedure should be added to the proposed design plan. Each added procedure is also checked if it requires any other procedures through checking its list of intermediate inputs. This continues until all required procedures are identified and added. After that, the best alternative procedure and its required procedures substitute the procedure under study in the proposed design plan.

5.2.2. Postmapping repair

If the case of the best alternative is not a full match case to the new problem, then there must be postmapping repair to fix the minimum remaining inconsistencies. This can be done by forwarding the mapped procedures to be repaired by the user with some suggestions about how they would be repaired. As an example, the output might be information about if a part is a special case of another and what extra attributes are not needed and suggests that these attributes be removed or be assigned neutral values.

Postmapping repair can also be done by changing the subdesign plan that contains an inconsistent procedure. In this case, the other similar cases will be searched again for subdesign plan alternatives that perform the same function as the one under study. Before any subdesign plan is considered an alternative, it must satisfy the pre-evaluation constraints relevant to the class of engineering design to which the new problem belongs. If alternatives are found, the best one can be chosen by comparing the contents of all input groups of all the procedures in this subdesign plan to their respective ones in the new problem. If a better subdesign plan is found, then it substitutes the old one in the proposed design plan. As a result of this substitution, all the new added procedures are checked for any inconsistencies.

5.2.3. Pre-evaluation constraints

Pre-evaluation constraints are relevant to a class of engineering design and can be checked for violation without any need for evaluating the design plan or any part of it. The

https://doi.org/10.1017/S0890060498125052 Published online by Cambridge University Press

purpose of these constraints is to maintain consistency and integration among all design components in an engineering system. As an example from structural engineering, a constraint may be added to maintain that a steel-base plate be added whenever a steel column is connected to a concrete foundation.

The relevant pre-evaluation design constraints are checked before mapping any part to the proposed design plan. If a suggested alternative violates any one of these constraints, then it should be excluded from the list of alternatives.

5.3. Reflecting the mapping on the new problem specification

Once all required procedures are mapped to the proposed design plan and all inconsistencies are repaired, the mapping process is considered complete. However, output attributes that were unknown to the user at the time of defining the problem still need to be added to the problem specification to complete the other adaptation processes; namely the evaluation, the verification, and the modification processes. The output list of each procedure in the proposed design plan is searched for any extra output attributes to add these attributes to their correspondent groups in the problem specification. After doing this, the proposed design plan is considered a potential design plan of the new problem and the mapping process then forwards this design plan for further processing by the other adaptation processes.

6. THE EVALUATION PROCESS

The potential design plan generated by the mapping process can not be considered a final design plan until it is evaluated. There is always a chance of hidden problems within the potential design plan. The main purpose of the evaluation process is to catch such imperfections and repair them. These imperfections may be in the form of inconsistencies between the type or the value of an attribute that is defined in the problem specification and the way it is used in the potential design plan. The other form of imperfection is a postevaluation constraint violation by any part of the design plan. The values of some output attributes are not known until the design plan is evaluated. Some of these attributes may be involved as elements of some constraints. As a result, these constraints can not be checked until the values of these attributes are known and violations cannot be reported until the evaluation process is complete. This kind of constraint violation is defined in the methodology as a postevaluation constraint violation versus the pre-evaluation constraint violation discussed in previous sections. The output of the evaluation process is an evaluation report that is a list that contains the values of the output attributes. This report is forwarded to the verification process to check for postevaluation constraint violations before reflecting the values of these attributes on the problem specification.

7. THE VERIFICATION PROCESS

The verification process uses the values of output attributes reported in the evaluation report in addition to the values of input attributes assigned by the user or the designer in checking the provided library of design constraints relevant to the class of engineering design under study. The constraints defined by the user in the problem specification are also checked in the same way as for the library constraints to report any postevaluation constraint violations. An example of postevaluation violation is that the computed thickness of a slab after selecting it as a ribbed slab makes the clearance of the story less than the minimum required limit.

The output of the verification process is a verification report that contains all the found violations. For each violation, the relevant constraint and the values of all attributes involved in this constraint are listed. This report is then forwarded to the modification process to make the required postevaluation repair.

It is worth noting that the verification process, evaluating constraints, as described here could be unbounded and complex for large problems. Handling this type of problems is beyond the scope of this paper. The reader is advised to check the CSP, Constraint Satisfaction Problem, literature.

8. THE MODIFICATION PROCESS

The modification process main task is to perform any required repair in the potential design plan that arose during or after the process of evaluation and was reported by the verification process. Repair, in most cases, is fixing the postevaluation constraint violations. For each violation reported in the verification report, the modification process separates the arguments of the constraint that are output attributes from the ones that are input attributes. After doing this for all reported violations, the modification process then turns to the procedures in the potential design plan. The lists of intermediate inputs and outputs of each procedure are then checked to determine if they contain any attributes involved in any violation. The modification process then applies the following heuristic rules:

- *rule1:* Choose the procedure that has the largest number of output attributes involved in postevaluation violations and that occur as intermediate inputs to other procedures. The resulted procedure(s) from applying this rule is/are the one(s) that if adjusted will affect a number of output attributes more than any other procedure would do.
- *rule2:* If applying *rule1* identifies in choosing more than one procedure, choose the procedure that occurs earlier in the potential design plan. This decision also helps in changing the values of a larger number of output attributes.

After applying *rule2*, the user is prompted to make adjustment to the resulted procedure. These adjustments should

https://doi.org/10.1017/S0890060498125052 Published online by Cambridge University Press

Table 2. *Design case for a short column with a circular cross section and resisting an axial load*

Case name:	COLUMN-1		
Goals:	Resist an axial load by a short column		
Input groups:			
LOADS:	float	axial-load	*N*
ENDS:	char	lower-end	*"braced"*
	char	upper-end	*"braced"*
GEOMETRY:	float	column-length	*l*
CROSS-SECTION:	char	cross-section-type	*"circular"*
	int	cross-section-diameter	*D*
BUILDING-MATERIAL:	char	building-material-type	*"reinforced concrete"*
	float	concrete-characteristic-strength	*Fcu*
	float	steel-characteristic-strength	*Fy*
Output groups:			
STRESSES:	float	ultimate-load	*N′*
CROSS-SECTION:	float	slenderness-factor	*s*
	float	longitudinal-steel-area	*As*
	float	cross-section-area	*A*
GEOMETRY:	float	effective-length	*le*
Constraints:	None		
Design plan:			
CALCULATING-EFFECTIVE-LENGTH:			
Inputs:		column-length	group:GEOMETRY
		lower-end	group:ENDS
		upper-end	group:ENDS
Intermediate inputs:		none	
Outputs:		effective-length	group:GEOMETRY
Instructions:			
		look up λ from table	
		effective-length = column-length $*\lambda$	
CALCULATING-SLENDERNESS-FACTORS:			
Inputs:		cross-section-diameter	group:CROSS-SECTION
Intermediate inputs:		effective-length	procedure: CALCULATING-EFFECTIVE-LENGTH
Outputs:		slenderness-factor	group:CROSS-SECTION
Instructions:			
		calculate slenderness-factor from a given relation	
CALCULATING-CROSS-SECTION-PROPERTIES:			
Inputs:		cross-section-diameter	group:CROSS-SECTION
Intermediate inputs:		None	
Outputs:		cross-section-area	group:CROSS-SECTION
Instructions:			
		cross-section-area= $\pi *$ (cross-section-diameter)^2	
CALCULATING-ULTIMATE-LOAD:			
Inputs:		axial-load	group:LOADS
Intermediate inputs:		None	
Outputs:		ultimate-load	group:STRESSES
Instructions:			
.....			
CALCULATING-LONGITUDINAL-STEEL:			
Inputs:		axial-load	group:LOADS
		characteristic-concrete-strength	group:BUILDING-MATERIAL
		characteristic-steel-strength	group:BUILDING-MATERIAL
Intermediate inputs:		ultimate-load	procedure: CALCULATING-ULTIMATE-LOAD
		cross-section-area	procedure: CALCULATING-SECTION-PROPERTIES
Outputs:		longitudinal-steel-area	group:CROSS-SECTION
Instructions:			
.....			

be done to satisfy the constraints that have direct relations to the output of this procedure. After making such adjustments, the resulted new potential design plan is forwarded to the evaluation process to be re-evaluated again.

9. EXAMPLES

The examples in this section are taken from the structural-component design domain. Component design usually does

https://doi.org/10.1017/S0890060498125052 Published online by Cambridge University Press

Table 3. *Design case for a short column resisting an axial load and a biaxial moment*

Case name:	COLUMN-2		
Goals:	Resist an axial load and a biaxial bending moment by a short column		
Input groups:			
LOADS:	float	axial-load	N
	float	x-bending-moment	M_x
	float	y-bending-moment	M_y
ENDS:	char	lower-end	*"braced"*
	char	upper-end	*"braced"*
GEOMETRY:	float	column-length	l
CROSS-SECTION:	char	cross-section-type	*"rectangular"*
	int	cross-section-height	h
	int	cross-section-width	b
BUILDING-MATERIAL:	char	building-material-type	*"reinforced concrete"*
	float	concrete-characteristic-strength	Fcu
	float	steel-characteristic-strength	Fy
Output groups:			
STRESSES:	float	ultimate-moment	M'
CROSS-SECTION:	float	x-slenderness-factor	s_x
	float	y-slenderness-factor	s_y
	float	longitudinal-steel-area	As
	float	cross-section-area	A
GEOMETRY:	float	effective-length	le
Constraints:	None		
Design plan:			
CALCULATING-EFFECTIVE-LENGTH:			
Inputs:		column-length	group:GEOMETRY
		lower-end	group:ENDS
		upper-end	group:ENDS
Intermediate inputs:		none	
Outputs:		effective-length	group:GEOMETRY
Instructions:			
.....			
CALCULATING-SLENDERNESS-FACTORS:			
Inputs:		cross-section-height	group:CROSS-SECTION
		cross-section-width	group:CROSS-SECTION
Intermediate inputs:		effective-length	procedure: CALCULATING-EFFECTIVE-LENGTH
Outputs:		x-slenderness-factor	group:CROSS-SECTION
		y-slenderness-factor	group:CROSS-SECTION
Instructions:			
.....			
CALCULATING-CROSS-SECTION-PROPERTIES:			
Inputs:		cross-section-height	group:CROSS-SECTION
		cross-section-width	group:CROSS-SECTION
Intermediate inputs:		None	
Outputs:		cross-section-area	group:CROSS-SECTION
Instructions:			
.....			
CONVERTING-BIAXIAL-BENDING-TO-UNIAXIAL:			
Inputs:		x-bending-moment	group:LOADS
		y-bending-moment	group:LOADS
Intermediate inputs:		None	
Outputs:		ultimate-moment	group:STRESSES
Instructions:			
.....			
CALCULATING-LONGITUDINAL-STEEL:			
Inputs:		axial-load	group:LOADS
		characteristic-concrete-strength	group:BUILDING-MATERIAL
		characteristic-steel-strength	group:BUILDING-MATERIAL
Intermediate inputs:		ultimate-moment	procedure: CALCULATING-ULTIMATE-LOAD
		cross-section-area	procedure: CALCULATING-SECTION-PROPERTIES
Outputs:		longitudinal-steel-area	group:CROSS-SECTION
Instructions:			
.....			

https://doi.org/10.1017/S0890060498125052 Published online by Cambridge University Press

Table 4. *Design case for a short-cased column resisting an axial load and a biaxial moment*

Case name:	COLUMN-3		
Goals:	Resist an axial load and a biaxial bending moment by a short column		
Input groups:			
LOADS:	float	axial-load	N
	float	x-bending-moment	M_x
	float	y-bending-moment	M_y
ENDS:	char	lower-end	*"braced"*
	char	upper-end	*"braced"*
GEOMETRY:	float	column-length	l
CROSS-SECTION:	char	cross-section-type	*"rectangular"*
	int	cross-section-height	h
	int	cross-section-width	b
	float	steel-section-area	Ass
BUILDING-MATERIAL:	char	building-material-type	*"steel+reinforced concrete"*
	float	concrete-characteristic-strength	Fcu
	float	steel-characteristic-strength	Fy
	float	steel-section-compression-strength	Pc
Output groups:			
STRESSES:	float	ultimate-moment	M'
CROSS-SECTION:	float	x-slenderness-factor	s_x
	float	y-slenderness-factor	s_y
	float	longitudinal-steel-area	As
	float	cross-section-area	A
GEOMETRY:	float	effective-length	le
Constraints:	None		
Design plan:			
CALCULATING-EFFECTIVE-LENGTH:			
Inputs:		column-length	group:GEOMETRY
		lower-end	group:ENDS
		upper-end	group:ENDS
Intermediate inputs:		none	
Outputs:		effective-length	group:GEOMETRY
Instructions:			
.....			
CALCULATING-SLENDERNESS-FACTORS:			
Inputs:		cross-section-height	group:CROSS-SECTION
		cross-section-width	group:CROSS-SECTION
Intermediate inputs:		effective-length	procedure: CALCULATING-EFFECTIVE-LENGTH
Outputs:		x-slenderness-factor	group:CROSS-SECTION
		y-slenderness-factor	group:CROSS-SECTION
Instructions:			
.....			
CALCULATING-CROSS-SECTION-PROPERTIES:			
Inputs:		cross-section-height	group:CROSS-SECTION
		cross-section-width	group:CROSS-SECTION
		steel-section-area	group:CROSS-SECTION
Intermediate inputs:		None	
Outputs:		cross-section-area	group:CROSS-SECTION
Instructions:			
.....			
CONVERTING-BIAXIAL-BENDING-TO-UNIAXIAL:			
Inputs:		x-bending-moment	group:LOADS
		y-bending-moment	group:LOADS
Intermediate inputs:		None	
Outputs:		ultimate-moment	group:STRESSES
Instructions:			
.....			
CALCULATING-LONGITUDINAL-STEEL:			
Inputs:		axial-load	group:LOADS
		characteristic-concrete-strength	group:BUILDING-MATERIAL
		characteristic-steel-strength	group:BUILDING-MATERIAL
Intermediate inputs:		ultimate-moment	procedure: CALCULATING-ULTIMATE-LOAD
		cross-section-area	procedure: CALCULATING-SECTION-PROPERTIES
Outputs:		longitudinal-steel-area	group:CROSS-SECTION
Instructions:			
.....			

https://doi.org/10.1017/S0890060498125052 Published online by Cambridge University Press

Table 5. *Problem specification for a short column resisting an axial load and a uniaxial moment*

Goals:	Resist an axial load and a uniaxial bending moment by a short column		
Input groups:			
LOADS:	float	axial-load	*N*
	float	x-bending-moment	M_x
ENDS:	char	lower-end	*"braced"*
	char	upper-end	*"braced"*
GEOMETRY:	float	column-length	*l*
CROSS-SECTION:	char	cross-section-type	*"rectangular"*
	int	cross-section-height	*h*
	int	cross-section-width	*b*
BUILDING-MATERIAL:	char	building-material-type	*"reinforced concrete"*
	float	concrete-characteristic-strength	*Fcu*
	float	steel-characteristic-strength	*Fy*
Output groups:			
CROSS-SECTION:	float	longitudinal-steel-area	*As*
Constraints:	None		

not provide a practical environment for demonstrating CBR techniques because the problem of designing a component is usually subject to a limited number of constraints. However, the structural-component design domain has been chosen because it provides a quick and easy environment to test and demonstrate theoretically the features of the methodology.

The knowledge base used for the examples consists of three cases for short-braced columns. The first case, shown in Table 2, is for a reinforced concrete column with a circular cross section supporting an axial load. The second case, shown in Table 3, is for a reinforced concrete column with a rectangular cross section. This column supports an axial load and is subjected to a biaxial bending moment. The third case, shown in Table 4, is a cased column with a rectangular cross section. This column also supports an axial load and is subjected to a biaxial bending moment.

9.1. Example 1

The first example is to design a short-braced reinforced concrete column required to support an axial load and is subjected to a uniaxial bending moment. The new problem specification is shown in Table 5.

Let us assume that the case selection process has proposed COLUMN-2 and COLUMN-3 as two similar cases with COLUMN-2 as the most similar case. The mapping module begins its work by considering the design plan of the most similar case, COLUMN-2, as a proposed design plan for the new problem. The involved input groups in the first procedure in the proposed design plan, CALCULATING-EFFECTIVE-LENGTH, are GEOMETRY and ENDS. The contents of each one of these groups are identical in the new problem and in COLUMN-2 case. Thus, this procedure is kept in the design plan without any need for further processing. Following the same steps for the next two procedures, CALCULATING-SLENDERNESS-FACTORS and CALCULATING-CROSS-SECTION-PROPERTIES, they are also kept in the proposed design plan because they are consistent with the new problem specification.

The fourth procedure, CONVERTING-BIAXIAL-BENDING-TO-UNIAXIAL has inputs from input group LOADS. The contents of this group in COLUMN-2 case are different from its contents in the new problem (no y-bending-moment in the new *problem*), thus defining a relation of *kind2*. That means that this part of the proposed design plan is inconsistent with the new problem specification. Because there is no alternative in the other similar case, COLUMN-3, the user is prompted to fix this part. The user is assumed to do the correct decision in this case that is deleting this procedure from the proposed design plan because there is no need for it in the new problem because the bending moment is already uniaxial.

The fifth procedure, CALCULATING-LONGITUDINAL-STEEL, has inputs from two input groups, LOADS and BUILDING-MATERIAL. As for the previous procedure, the contents of input group LOADS in the new problem are different from its contents in COLUMN-2 making the current definition of this procedure inconsistent with the new problem because there is no bending moment in the Y direction. The process of searching for alternatives begins and an alternative procedure for calculating longitudinal steel is found in COLUMN-3. What is needed now is to determine whether this alternative procedure is better than the one in the proposed design plan *that* belongs to COLUMN-2 case. The contents of input group LOADS in the new problem define a special case of its contents in both COLUMN-2 and COLUMN-3. Thus, the input group LOADS in both cases has a relation of *kind2*. The contents of input group BUILDING-MATERIAL in the new problem are identical to its contents in COLUMN-2 case and different from its

https://doi.org/10.1017/S0890060498125052 Published online by Cambridge University Press

Table 6. *Design case for a short column resisting an axial load and a uniaxial moment*

Case name:	COLUMN-4		
Goals:	Resist an axial load and a uniaxial bending moment by a short column		
Input groups:			
LOADS:	float	axial-load	N
	float	x-bending-moment	M_x
ENDS:	char	lower-end	*"braced"*
	char	upper-end	*"braced"*
GEOMETRY:	float	column-length	l
CROSS-SECTION:	char	cross-section-type	*"rectangular"*
	int	cross-section-height	h
	int	cross-section-width	b
BUILDING-MATERIAL:	char	building-material-type	*"reinforced concrete"*
	float	concrete-characteristic-strength	Fcu
	float	steel-characteristic-strength	Fy
Output groups:			
STRESSES:	float	ultimate-moment	M'
CROSS-SECTION:	float	x-slenderness-factor	s_x
	float	y-slenderness-factor	s_y
	float	longitudinal-steel-area	As
	float	cross-section-area	A
GEOMETRY:	float	effective-length	le
Constraints:	None		
Design plan:			
CALCULATING-EFFECTIVE-LENGTH:			
Inputs:		column-length	group:GEOMETRY
		lower-end	group:ENDS
		upper-end	group:ENDS
Intermediate inputs:		none	
Outputs:		effective-length	group:GEOMETRY
Instructions:			
.....			
CALCULATING-SLENDERNESS-FACTORS:			
Inputs:		cross-section-height	group:CROSS-SECTION
		cross-section-width	group:CROSS-SECTION
Intermediate inputs:		effective-length	procedure: CALCULATING-EFFECTIVE-LENGTH
Outputs:		x-slenderness-factor	group:CROSS-SECTION
		y-slenderness-factor	group:CROSS-SECTION
Instructions:			
.....			
CALCULATING-CROSS-SECTION-PROPERTIES:			
Inputs:		cross-section-height	group:CROSS-SECTION
		cross-section-width	group:CROSS-SECTION
Intermediate inputs:		None	
Outputs:		cross-section-area	group:CROSS-SECTION
Instructions:			
.....			
CALCULATING-LONGITUDINAL-STEEL:			
Inputs:		axial-load	group:LOADS
		x-bending-moment	group:LOADS
		characteristic-concrete-strength	group:BUILDING-MATERIAL
		characteristic-steel-strength	group:BUILDING-MATERIAL
Intermediate inputs:		cross-section-area	procedure: CALCULATING-SECTION-PROPERTIES
Outputs:		longitudinal-steel-area	group:CROSS-SECTION
Instructions:			
.....			

contents in COLUMN-3 case. Thus, this group has a relation of *kind1* in COLUMN-2 case and a relation of *kind4* in COLUMN-3 case. By applying *rule-1*, *rule-2*, and *rule-3* defined in Section 5.2.1, the procedure in COLUMN-2 case is found better than the alternative in COLUMN-3 case. By checking the list of intermediate inputs of this procedure, the systems finds that it requires the presence of procedure CALCULATING-CROSS-SECTION-PROPERTIES, which

https://doi.org/10.1017/S0890060498125052 Published online by Cambridge University Press

is already in the design plan. However, although the procedure in COLUMN-2 case is the best available procedure, it is still inconsistent with the new problem. The user is prompted to fix this part with the suggestion of assigning neutral values or removing the undefined attribute, y-bending-moment, because input group LOADS in the new problem defines a special case of what is used in this procedure. The user is assumed to do the correct fixing.

By now, the proposed design plan has become a potential design plan. The outputs of each procedure are added to the new problem specification and the design plan is forwarded to the evaluation module. The potential design plan is then evaluated. The verification process is assumed here to report no violations in postevaluation constraints. That makes the potential design plan a final design plan. Finally, the new problem and its design plan are forwarded to the case-storage process to be added to the knowledge base as a new design case under the name of COLUMN-4 as shown in Table 6.

9.2. Example 2

The second example is to design a short-braced, reinforced-concrete column with a rectangular cross section required to support an axial load. The new problem specification is shown in Table 7.

Let us assume that two cases are proposed by the case selection process as similar cases. These two cases are COLUMN-4 and COLUMN-2, with COLUMN-4 case as the most similar case. The mapping module begins by taking the design plan of COLUMN-4 case as a proposed design plan for the new problem. The first three procedures in the proposed design plan, CALCULATING-EFFECTIVE-LENGTH, CALCULATING-SLENDERNESS-FACTORS, and CALCULATING-CROSS-SECTION-PROPERTIES have inputs from groups whose contents in the new problem are the same as their contents in COLUMN-4 case. So these three procedures are kept in the proposed design plan without any change. The fourth procedure, CALCULATING-LONGITUDINAL-STEEL, has inputs from groups LOADS and BUILDING-MATERIAL. The contents of input group LOADS in the new problem are different from its contents in COLUMN-4 case; so this procedure is considered inconsistent with the new problem. When searching for alternatives, COLUMN-1 case is proposed because the contents of each one of input groups, LOADS and BUILDING-MATERIAL, in COLUMN-1 case are the same as in the new problem. Thus, the procedure in COLUMN-1 case is a better alternative and should be mapped to the proposed design plan. By checking the list of intermediate inputs of this procedure in COLUMN-1 case, the system finds that this procedure requires the presence of procedure CALCULATING-ULTIMATE-LOAD, which is not in the proposed design plan. This procedure has inputs from input group BUILDING-MATERIAL, which defines a full match case between COLUMN-1 case and the new problem. The list of intermediate inputs of this procedure is empty so no more procedures are required. The two procedures, CALCULATING-ULTIMATE-LOAD and CALCULATING-LONGITUDINAL-STEEL from COLUMN-1 case then substitute the procedure CALCULATING-LONGITUDINAL-STEEL from COLUMN-4 case in the proposed design plan. All mapped procedures are consistent with the new problem specification so no postmapping repair is needed. The outputs of the procedures are then added to the new problem specification, and the proposed design plan has become a potential design plan.

By evaluating this design plan and verifying the results, assuming that there *are* no postevaluation constraint violations, the potential design plan is accepted, and the new problem with its design plan is forwarded to the case-storage process, to be added to the knowledge base as a new case under the name COLUMN-5 as shown in Table 8.

Table 7. *Problem specification for a short column resisting an axial load*

Goals:	Resist an axial load by a short column		
Input groups:			
LOADS:	float	axial-load	*N*
ENDS:	char	lower-end	*“braced”*
	char	upper-end	*“braced”*
GEOMETRY:	float	column-length	*l*
CROSS-SECTION:	char	cross-section-type	*“rectangular”*
	int	cross-section-height	*h*
	int	cross-section-width	*b*
BUILDING-MATERIAL:	char	building-material-type	*“reinforced concrete”*
	float	concrete-characteristic-strength	*Fcu*
	float	steel-characteristic-strength	*Fy*
Output groups:			
CROSS-SECTION:	float	longitudinal-steel-area	*As*
Constraints:	None		

https://doi.org/10.1017/S0890060498125052 Published online by Cambridge University Press

Table 8. *Design case for a short column with a rectangular cross section and resisting an axial load*

Case name:	COLUMN-5		
Goals:	Resist an axial load by a short column		
Input groups:			
LOADS:	float	axial-load	*N*
ENDS:	char	lower-end	*"braced"*
	char	upper-end	*"braced"*
GEOMETRY:	float	column-length	*l*
CROSS-SECTION:	char	cross-section-type	*"rectangular"*
	int	cross-section-height	*h*
	int	cross-section-width	*b*
BUILDING-MATERIAL:	char	building-material-type	*"reinforced concrete"*
	float	concrete-characteristic-strength	*Fcu*
	float	steel-characteristic-strength	*Fy*
Output groups:			
STRESSES:	float	ultimate-load	*N′*
CROSS-SECTION:	float	x-slenderness-factor	s_x
	float	y-slenderness-factor	s_y
	float	longitudinal-steel-area	*As*
	float	cross-section-area	*A*
GEOMETRY:	float	effective-length	*le*
Constraints:	None		
Design plan:			
CALCULATING-EFFECTIVE-LENGTH:			
Inputs:		column-length	group:GEOMETRY
		lower-end	group:ENDS
		upper-end	group:ENDS
Intermediate inputs:		none	
Outputs:		effective-length	group:GEOMETRY
Instructions:			
.....			
CALCULATING-SLENDERNESS-FACTORS:			
Inputs:		cross-section-height	group:CROSS-SECTION
		cross-section-width	group:CROSS-SECTION
Intermediate inputs:		effective-length	procedure: CALCULATING-EFFECTIVE-LENGTH
Outputs:		x-slenderness-factor	group:CROSS-SECTION
		y-slenderness-factor	group:CROSS-SECTION
Instructions:			
.....			
CALCULATING-CROSS-SECTION-PROPERTIES:			
Inputs:		cross-section-height	group:CROSS-SECTION
		cross-section-width	group:CROSS-SECTION
Intermediate inputs:		None	
Outputs:		cross-section-area	group:CROSS-SECTION
Instructions:			
.....			
CALCULATING-ULTIMATE-LOAD:			
Inputs:		axial-load	group:LOADS
Intermediate inputs:		None	
Outputs:		ultimate-load	group:STRESSES
Instructions:			
.....			
CALCULATING-LONGITUDINAL-STEEL:			
Inputs:		axial-load	group:LOADS
		characteristic-concrete-strength	group:BUILDING-MATERIAL
		characteristic-steel-strength	group:BUILDING-MATERIAL
Intermediate inputs:		ultimate-load	procedure: CALCULATING-ULTIMATE-LOAD
		cross-section-area	procedure: CALCULATING-SECTION-PROPERTIES
Outputs:		longitudinal-steel-area	group:CROSS-SECTION
Instructions:			
.....			

https://doi.org/10.1017/S0890060498125052 Published online by Cambridge University Press

10. CONCLUSIONS

A methodology for case-based adaptation in engineering design is described and implemented. The main goal of this methodology is to assist the process of adaptation as part of CBR design. The system is organized and implemented to assist and interact with the designer in each subprocess carried out by a submodule. This way, the user still controls the adaptation process instead of leaving this task to the system.

Among the issues addressed in developing this methodology is a case representation model as a flexible container of engineering experience. The adaptation process is divided into four processes: the mapping process, the evaluation process, the verification process, and the modification process.

By developing such a system and addressing the resultant issues, many gains have been achieved including :

- A case representation model that is flexible enough to represent engineering design problems from different fields or classes.
- An adaptation methodology that can be integrated within a CBR system or used separately in the domain of engineering design

REFERENCES

Bain, W. (1989). "JUDGE." In *Case-based Reasoning*, (Riesbeck, C.K. and Schank, R.C. (Eds.). Lawrence Erlbaum, Hillsdale, N.J.

Bardasz, T., & Zeid, I. (1991). Applying analogical problem solving to mechanical design. *CAD Journal 23*, 202–212.

Bardasz, T., & Zeid, I. (1993). DEJAVU: Case-based reasoning for mechanical design. ***AI****EDAM 7*, 111–124.

Barletta, R., & Hennessy, D. (1989). Case adaptation in autoclave layout design. *Proc. 1989 Case-based Reasoning Workshop*, pp. 203–207. Morgan Kaufmann, San Mateo, California.

Daube, F., & Hayes-Roth, B. (1990). A case-based mechanical redesign system, 1990. *AAAI Proceedings*, pp. 1402–1407.

Goel, A., & Chandrasekaran, B. (1989). Use of device model in adaptation of design cases. *Proc. 1989 Case-based Reasoning Workshop*, pp. 100–109. Morgan Kaufmann, San Mateo, California.

Hammond, K. (1986). Learning to anticipate and avoid planning problems through the explanation of failure. *Proc. Fifth National Conf. Artificial Intelligence 1986*, pp. 556–560. Morgan Kaufmann Publishers, Inc., Los Altos, California.

Hennessy, D.H., & Hinkle, D. (1992). Applying case-based reasoning to autoclave loading. *IEEE Expert 7*, 21–26.

Hinrichs, T.R. (1989). Strategies for adaption and recovery in a design problem solver. *Proc. 1989 Case-based Reasoning Workshop*, pp. 115–118. Morgan Kaufmann: San Mateo, California.

Kolodner, J. (1993). *Case-based reasoning*. Morgan Kaufmann, San Mateo, California.

Koton, P. (1988). Reasoning about evidence in causal reasoning. *Proc. Seventh National Conf. Artificial Intelligence 1988*, pp. 256–207. Morgan Kaufmann, San Matteo, California.

Maher, M.L., Balachandran, M.B., & Zhang, D.M. (1995). *Case-based reasoning in design*. Lawrence Erlbaum, Mahwan, NJ.

Maher, M.L., & Zhang, D.M. (1993). CADSYN: A case-based design process model. ***AI****EDAM 7*, 97–110.

Navinchandra, D. (1988). Case-based Reasoning in CYCLOP, a design problem solver. *Proc. 1988 Case-based Reasoning Workshop*, pp. 286–301. Morgan Kaufmann: San Mateo, California.

Roderman, S., & Tsatsoulis, C. (1993). PANDA: A case-based system to AI novice designers. ***AI****EDAM 7*, 125–133.

Shinn, H.S. (1988). Abstractional analogy: A model of analogical reasoning. *Proc. 1988 Case-based Reasoning Workshop*, pp. 370–387. Morgan Kaufmann: San Mateo, California.

Shiva Kumar, H., & Krishnamoorthy, C.S. (1995). A framework for case-based reasoning in engineering design. ***AI****EDAM 9*, 161–182.

https://doi.org/10.1017/S0890060498125052 Published online by Cambridge University Press