INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

ProQuest Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

Case-Based Reasoning for Modeling Bridge Deterioration

George Morcous

A Thesis

In

The Department

Of

Building, Civil, and Environmental Engineering

Presented in Partial Fulfillment of the Requirements
For the Degree of Doctor of Philosophy at
Concordia University
Montreal, Quebec, Canada

November 2000

ABSTRACT

Case-Based Reasoning for Modeling Bridge Deterioration

George Morcous, Ph.D.
Concordia University, 2000

Transportation infrastructure, especially highway bridges, is characterized by their growing deterioration rate. Increased traffic loads, severity of environmental conditions, and deferred maintenance decisions are the main reasons for this rapid deterioration. Transportation agencies have developed Bridge Management Systems (BMS's) to select optimal maintenance actions and to predict future funding needs. BMS's require bridge deterioration models to predict the future condition of bridges and to estimate their remaining service life. Most of the state-of-the-art BMS's employ Markov Decision Process (MDP) in modeling bridge deterioration. The main shortcomings of Markovian models are that they assume state independence, neglect improvement actions taken in the past, and overlook the interaction of bridge components. Moreover, these models are difficult to update when new data are obtained or a different rating system is used.

In this research, a Case-Based Reasoning (CBR) approach was introduced to provide BMS's with a realistic, accurate, and generic deterioration model that eliminates the previous shortcomings. This approach predicts the future condition of a query bridge case by retrieving bridge cases stored in the BMS database that are similar to the query case in their physical features, environmental and operating conditions, and previous inspection and maintenance records.

To implement this approach, a CBR shell called CBRMID was developed to be used in modeling the deterioration of any infrastructure facility. This shell was designed to meet a set of requirements, such as hierarchical decomposition of cases, representation of time-dependent data, data accumulation, case adaptation, data derivation, system modularity, versatility, and extensibility. CBRMID consists of three main modules: (i) Case Based Reasoner, for case retrieval and adaptation; (ii) System Maintainer, for case and knowledge accumulation and modification; and (iii) Database, for case and knowledge representation. CBRMID was implemented using the object-oriented programming language C++, the object-oriented database management system ObjectStore 6.0, and the expert system programming language CLIPS 6.10.

Cases obtained from the Ministry of Transportation of Quebec (MTQ) database were employed in developing a "proof of concept" CBR application for modeling the deterioration of concrete bridge decks. Comparison between the solutions provided by the developed application and the actual solutions of real-world cases indicated that the CBR approach has a great potential in modeling bridge deterioration and, consequently, may have a serious impact on our transportation system.

ACKNOWLEDGEMENT

I would like to acknowledge my supervisors Dr. A. M. Hanna and Dr. H. Rivard for their tremendous effort in advising, teaching, and supporting me throughout this research.

I would like to thank Dr. S. Alkass, who was a co-supervisor at the initial stage of my Ph.D. program leading to my Ph.D. proposal (from Sept. 97 to Apr. 99).

Also, I would like to thank Eng. Guy Richard and Eng. René Gagnon in the structures department of the Ministry of Transportation of Quebec and the bridge engineers in the structural office of the Ministry of Transportation of Ontario for their fruitful cooperation and great help in providing me with knowledge, data, and manuals.

Constructive consultations with Dr. Z. Khalil concerning data analysis and with Mrs. M. O' Malley concerning technical writing are appreciated.

The financial support received from Concordia University fellowship and the National Sciences and Engineering Research Council of Canada are gratefully acknowledged.

Finally, I am greatly thankful to my wife, family, and friends for their continuous encouragement and extreme love.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

Abbreviation	Description
AASHTO	American Association of State Highway and Transportation Official
ADT	Average Daily Traffic
AI	Artificial Intelligence
ANN	Artificial Neural Network
ARU	Add, Remove, and Update
BMS	Bridge Management System
CBR	Case-Based Reasoning
CBRMID	Case-Based Reasoning for Modeling Infrastructure Deterioration
CDF	Cumulative Distribution Function
CLIPS	C Language Integrated Production System
DB	Database
DBMS	Database Management System
DLL	Dynamic Library Link
DOT	Department of Transportation
ES	Expert System
EXE	Executable
FHWA	Federal Highway Administration
GUI	Graphical User Interface
HBRRP	Highway Bridge Replacement and Rehabilitation Program
ICS	Indice Combiné d'une Structure
IDE	Integrated Development Environment
IMS	Infrastructure Management System
IR	Information Retrieval
ISTEA	Intermodal Surface Transportation Efficiency Act.
LMDP	Latent Markov Decision Process
MCR	Material Condition Rating
MDP	Markov Decision Process
MFC	Microsoft Foundation Classes
MLE	Maximum Likelihood Estimation
MR&R	Maintenance, Rehabilitation, and Replacement
MTO	Ministry of Transportation of Ontario
MTQ	Ministry of Transportation of Quebec
NBI	National Bridge Inventory
NBIP	National Bridge Inspection Program
NCHRP	National Coperative Highway Research Program
NDT	Non-Destructive Testing

ODBC	Object Database Connectivity
OODBMS	Object-Oriented Database Management System
OOP	Object-Oriented Programming
OSIMS	Ontario Structure Inspection Management System
OSIS	Ontario Structural Inventory System
PCI	Pavement Condition Index
PCR	Performance Condition Rating
PMS	Pavement Management System
RDBMS	Relational Database Management System
RDD	Rapid Database Development
SBRP	Special Bridge Replacement Program
SGS	Système de Gestion des Structures
SI	Sufficiency Index
SQL	Structured Query Language
TFN	Triangular Fuzzy Number
UML	Unified Modeling Language
VC++	Microsoft Visual C++

CHAPTER 1

INTRODUCTION

1.1 BACKGROUND

A nation's infrastructure plays an essential role in the nation's economy, and preserving its condition is crucial for improving the economy strength. A nation's infrastructure comprises a wide range of facilities that provide the public with vital and indispensable services. These facilities can be categorized into transportation, building, water and wastewater, and telecommunication facilities. Transportation facilities are the most demanding item among all categories of infrastructure facilities. In the U.S., the budget of the fiscal year 2001 that was submitted to the Congress assigns $54.9 billion to be spent on transportation infrastructure (U.S. Government 2000). In Canada, government and business investments devoted to transportation infrastructure represent 20% of the total investments in the Canadian economy (Transport Canada 1998).

Transportation infrastructure facilities comprise highways and highway structures (mainly bridges), railways, marines, airports, etc. Figure 1.1 shows the distribution of the U.S. Department of Transportation's (DOT) budget for the fiscal year 2000 over its different administrations' needs (Slater 1998). It can be seen that the Federal Highway Administration (FHWA), which is concerned mainly with highways and bridges, consumes more than half (57%) of the U.S. DOT budget. Similarly, Figure 1.2 shows the distribution of transportation investments in Canada over different transportation modes

averaged from 1993 to 1996. It can be seen that road investments constitute 87% of the total transportation investments in Canada (Transport Canada 1998). This means that the greatest portion of the funding allocated to transportation infrastructure is assigned to highway and highway-bridge projects.

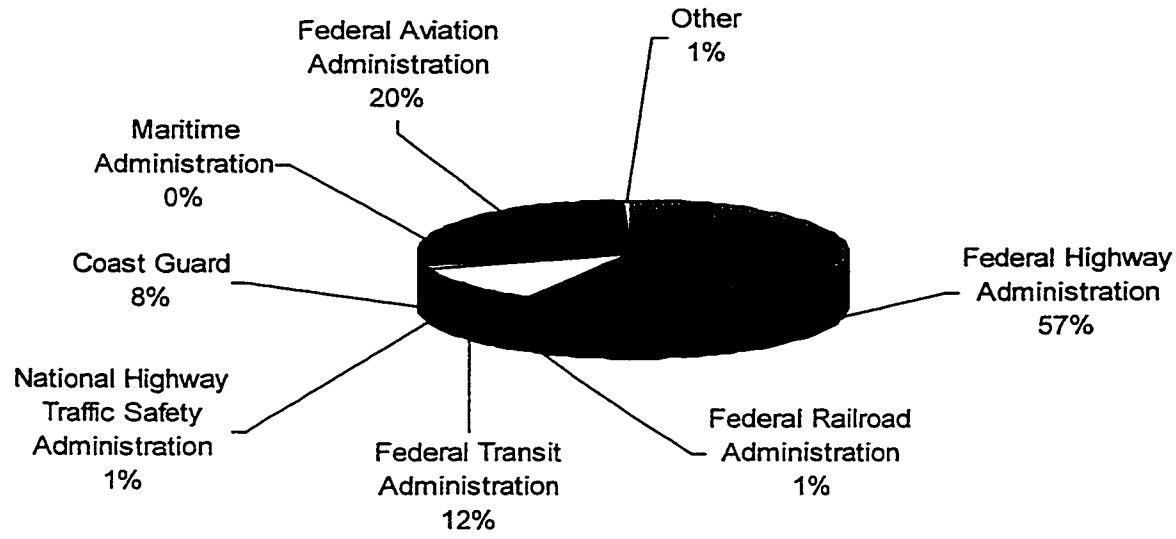


Figure 1.1: Distribution of the U.S. DOT budget for the fiscal year 2000 (Slater 1998)

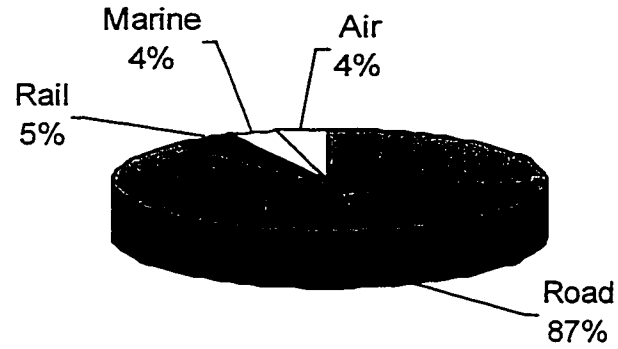


Figure 1.2 Distribution of transportation investments in Canada between 1993-1996 (Transport Canada 1998)

Although highways are the dominant element in any roadway network and their poor condition impairs the overall performance of the network, highway bridges are considered more critical and more vital links in any roadway network. This is because the full or partial failure of these links paralyses the overall performance of the network and may lead to serious catastrophes. Therefore, the particular concern of this research project is on highway bridges.

Highway bridges are characterized by their seriously and continuously deteriorating condition. As evidence, in 1998, the U.S. DOT revealed that about 30% of their 583,414 highway bridges were classified as either structurally deficient or functionally obsolete (U.S. DOT 1999). A structurally deficient bridge, according to the FHWA, is a bridge that has a restricted load-carrying capacity, requires immediate improvement to remain open, or is closed for services (FHWA 1995). A functionally obsolete bridge is one that may be structurally inadequate, has intolerable roadway geometry, has a narrow roadway width, or has a low under-clearance or over-clearance (Sobanjo 1991). Increasing traffic volume, excessive weight, speed, and size of modern vehicles, and severity of the environment conditions are the main reasons for this rapid deterioration of highway bridges. In addition, most of these bridges were constructed many years ago and have reached the end of their design life. In the U.S., about half the bridges were built before 1940 (Hudson et al. 1987). This means that they now have an average age of about 60 years. In Canada, about 40% of the bridges are over 35 years old (Lounis 1999). Even in newly built bridges, deterioration caused by service conditions and deferred maintenance is growing significantly.

Research emphasis in recent years has been shifting from the design and construction of new bridges to the maintenance, rehabilitation, and replacement (MR&R) of existing ones in order to upgrade the existing networks and to keep them in a safe and serviceable condition. MR&R activities demand a tremendous amount of funds that often exceed the available budget provided to transportation agencies, even in wealthy countries. In the U.S., the cost of MR&R activities for their 583,414 highway bridges is estimated at $90 billion (Aktan et al. 1996), while the federal funds assigned to bridge projects are estimated at $2.6 billion and distributed as shown in Figure 1.3 (FHWA 1997). In Canada, the deferred maintenance backlog is estimated at $10 billion (Lounis 1999). Therefore, bridge management systems (BMSs) have been developed to provide decision support in the allocation of the available limited funds and to facilitate the optimum selection of the most cost-effective maintenance actions on both project and network levels.

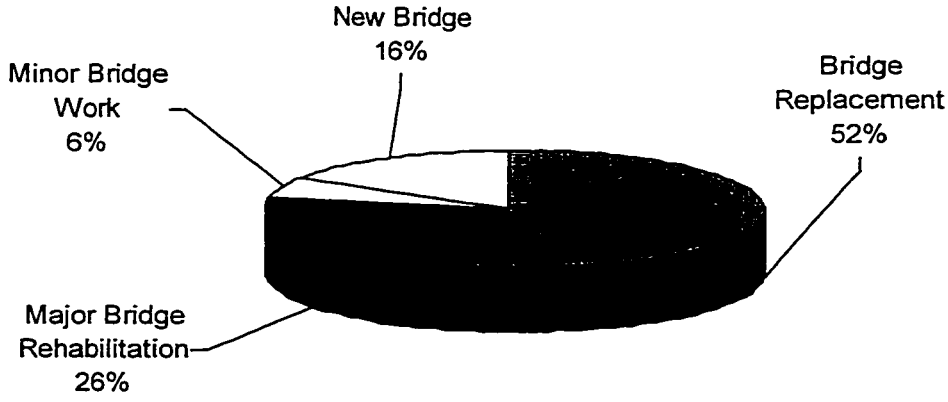


Figure 1.3: Distribution of U.S. federal funds for bridge projects from 1993 to 1997 (FHWA 1997).

By the mid 90's, almost all transportation agencies in North America had fully operational BMS's for managing the bridge network under their jurisdiction. At the project level, a BMS assists decision-makers in determining the optimal cost-effective maintenance actions for a specific bridge, while at the network level, such a decision is made for an entire network of bridges considering the budget constraints. Making these decisions without predicting the future condition of these bridges and without identifying their future funding needs restricts the benefits obtained from these BMS's. Therefore, bridge deterioration models have been introduced to predict the future condition of different bridge elements and to identify their future funding requirements. The American Association of State Highway and Transportation Officials (AASHTO) has prescribed the bridge deterioration model as one of the minimum requirements of any BMSs (AASHTO 1993). The present research focuses on this essential component in BMS's.

1.2 PROBLEM DEFINITION

Since the late 80's, extensive research efforts have been devoted to modeling bridge deterioration. However, a realistic, accurate, and generic bridge deterioration model has not been found yet. The variation in bridge data availability from one transportation agency to another (Turner and Richardson 1994), the dynamic nature of some types of bridge data (Kriviak 1999), the non-existence of unified inspection procedures, rating system, and inspection period (Turner and Richardson 1994), the multiplicity of bridge deterioration factors (Shahin 1994), the variability of deterioration mechanisms from one element to another (Sianipar and Adams 1997), the interaction among element deterioration mechanisms (Cesare et al. 1992), the latent nature of the deterioration

process (i.e. measuring deterioration indicators and not the process itself) (Bulusu and Sinha 1997), and the uncertainty and randomness of bridge deterioration (i.e. existence of unobserved deterioration factors) (Madanat et al. 1995) are the main difficulties that hinder modeling bridge deterioration accurately. As an evidence for how these difficulties affect modeling bridge deterioration, Figure 1.4 shows a dot plot of the material condition for 8252 bridge decks in Quebec against their ages in 1998. This Figure indicates that there is no clear or simple relationship that can model the change of deck condition with its age. This is due to the fact that many significant deterioration factors are neglected, the interactions among the deterioration mechanisms of bridge elements are overlooked, the latent nature and the randomness of the deterioration process are disregarded, and the previous maintenance and repair actions are not considered.

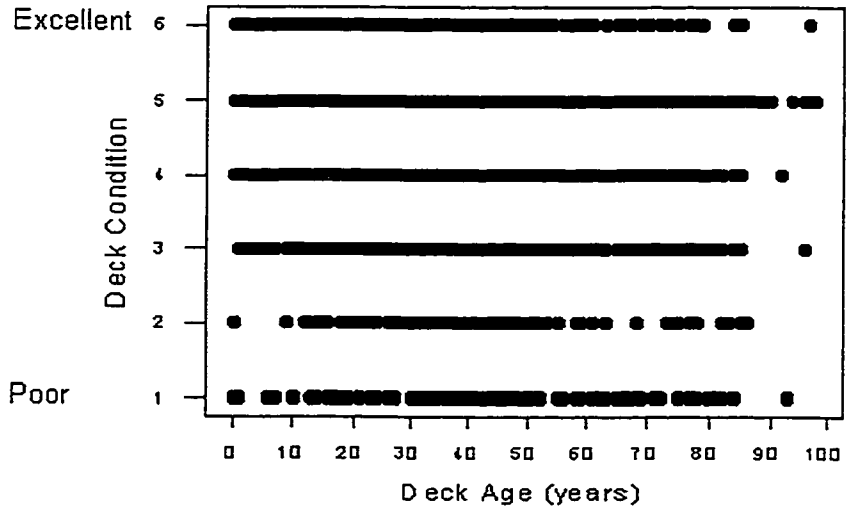


Figure 1.4: Dot plot of deck condition versus age for bridges in Quebec

Accordingly, efforts in this study are directed to develop a realistic accurate, and generic bridge deterioration model that overcomes most of the described difficulties and eliminates the shortcomings of the state-of-the-art bridge deterioration models.

1.3 THESIS ORGANIZATION

This study will be presented as follows:

- **Chapter 2** Literature Review: This chapter will present a brief discussion of the history of legislation concerning BMS's along with an overview of BMS modules and the major BMS's that are currently in use. Then, the definition, objective, and classification of bridge deterioration models will be provided. Mechanistic, deterministic, stochastic, and artificial intelligence deterioration models will be presented in detail along with a discussion of the shortcomings of each category. Research objectives and the scope of the study will conclude the chapter.
- **Chapter 3** Proposed Methodology: This chapter will introduce Case-Based Reasoning (CBR) systems, their aspects, and their different uses. The idea of using the CBR approach in modeling bridge deterioration will be explained. The requirements, resources, and functionality of the proposed CBR system will be given.
- **Chapter 4** System Analysis: This chapter will introduce a new method of decomposing bridges hierarchically. This decomposition identifies the different concepts of the bridge management domain and eliminates the shortfalls of the current decompositions. Then, the different categories of bridge data will be described and the conceptual model of bridges will be presented.
- **Chapter 5** System Design: This chapter will present the need for a CBR shell that satisfies the system requirements. The different modules of that shell along with the function of each module will be provided. Then, the part of the shell design model that concerns each of the four CBR aspects: case representation, case retrieval, case adaptation, and case accumulation, will be discussed in detail.

- **Chapter 6** System Implementation: This chapter will present the system implementation map that describes the developed software components and their dependencies. Then, an illustrative example, which demonstrates the developed prototype and its different capabilities, will be presented.
- **Chapter 7** System Testing and Evaluation: This chapter will present the development of a "proof of concept" CBR application for modeling bridge deck deterioration to verify the success of the system design and implementation. Solutions of actual cases will be used to validate the system performance. Finally, a comprehensive evaluation of the developed system will be provided.
- **Chapter 8** Summary, Contributions, and Recommendations: This chapter summarizes the present research work, highlights its contributions, and suggests recommendations for future research.

CHAPTER 2

LITERATURE REVIEW

2.1 BACKGROUND

The collapse of the Silver Bridge between Point Pleasant, WV, and Callipolis, OH, in 1967 and the loss of 47 lives due to the instantaneous fracture of an eye-bar turned great attention to the condition of highway bridges in the United States and in the whole world (Czepiel 1995). At that time, there were no systematic procedures anywhere for maintaining or even monitoring the condition of a network of bridges.

In 1968, the FHWA created the National Bridge Inspection Program (NBIP) to address the problem of safety in bridges (Czepiel 1995). This program mandated every DOT in the U.S. to keep track of the condition of the bridges under its jurisdiction. Data collected from applying the NBIP were submitted after every inspection cycle to the FHWA to be included in the National Bridge Inventory (NBI) database. In 1970, the data stored in the NBI were used as the basis for funding the Special Bridge Replacement Program (SBRP). This program provided federal funding to States in order to replace bridges that were close to failure. Later, the SBRP was replaced by the Highway Bridge Replacement and Rehabilitation Program (HBRRP), which provided funding for bridge rehabilitation in addition to replacement projects. The intent of this change was to begin repairing bridges before they deteriorated to a critical state.

In 1972, the FHWA published the Recording and Coding Guide for the Structure Inventory and Appraisal of the Nation's Bridges that outlined the specific elements that were required to be inspected and the inspection procedures (Czepiel 1995). In 1991, the Intermodal Surface Transportation Efficiency Act. (ISTEA) recognized the need for the preventive maintenance of bridges in order to minimize problems before they occur. In 1993, the ISTEA stipulated a legislation that every DOT in the U.S. must implement a management system for its network of bridges The deadline for having this system fully operational was Oct. 1, 1994.

2.2 BRIDGE MANAGEMENT SYSTEMS

A bridge management system is defined as a rational and systematic approach to organizing and carrying out all the activities related to maintaining a network of bridges. (Scherer and Glagola 1994). The major objective of a BMS is to assist bridge managers in making consistent and cost-effective decisions related to maintenance, rehabilitation, and replacement of bridges and in identifying future funding needs. These decisions can be optimized for an individual bridge project and/or for the entire network of bridges (Hudson 1987). The mandated requirements for a model BMS are outlined in the interim final rule that was issued by the ISTEA in 1993. This rule in conjunction with the AASHTO guidelines for BMS's, suggest that a model BMS should include four basic modules (Czepiel 1995): a database, a deterioration model, a cost model, and an optimization model. These modules are organized as shown in Figure 2.1 according to their dependencies (i.e. the arrows point to the dependent modules).

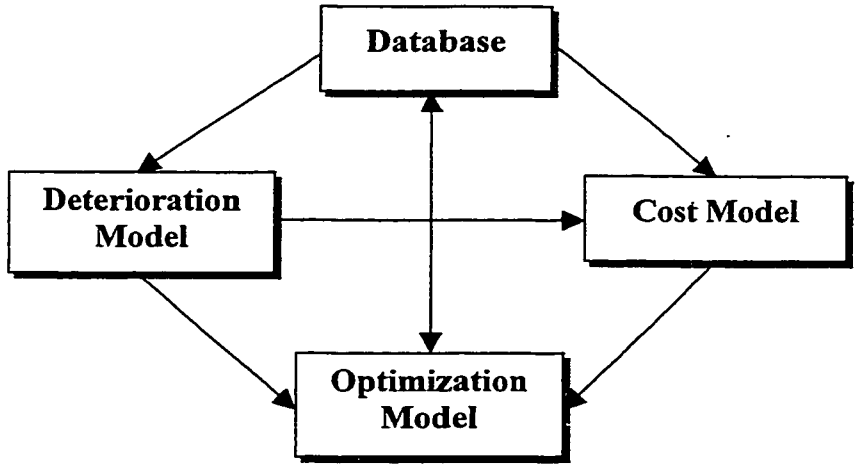


Figure 2.1: Modules of a model BMS (Czepiel 1995)

The database module is the heart of any BMS. This module involves mainly the collection and storage of inventory, inspection, appraisal, and maintenance data. These data are used by the other BMS modules as a basis for all decisions and actions analyzed by the BMS (Hudson 1987).

The deterioration model is defined as the link between measures of bridge condition and a vector of explanatory variables (Ben-akiva and Gopinath 1995). A measure of bridge condition in its simplest form is an assessment of the extent and the severity of a specific damage. More complex measures combine the extent and severity of different damage types in one index. The explanatory variables are defined as the factors affecting bridge deterioration and can be observed or measured. The function of this model is to predict the future condition of different bridge components. This prediction facilitates the determination of the optimal maintenance strategy and the estimation of future funding requirements. A detailed discussion of bridge deterioration models will be presented in the next section.

The cost model typically estimates two types of costs: agency cost and user cost (Johnston et al. 1994). Agency cost is estimated to determine the expenditures incurred (labor, materials, and equipment) to perform a specific maintenance action. User cost measures the impact of the deterioration on the user as a function of the bridge condition. User costs can also be generated due to bridge deficiencies such as narrow width, low clearance, poor alignment, and low load capacity.

The optimization model determines the least-cost MR&R strategies for different bridge components. This model may use life cycle cost analysis to account for the maintenance performed on a bridge throughout its entire service life. This analysis may suggest some maintenance actions to be made before the bridge condition worsens and the cost of maintaining it increases. A BMS may take either a top-down or bottom-up approach to optimization (Czepiel 1995). The top-down approach determines the desired goals for the entire network and then selects bridges that achieve these goals. The bottom-up approach determines the optimum action for each individual bridge and then selects actions that optimize the network condition. The bottom-up approach provides more meaningful results but it is slower with large bridge populations.

The two most popular BMS's are Pontis and BRIDGIT that have a generic design that can be adapted to accommodate the individual needs of most transportation agencies. Pontis is the predominant BMS employed in the U. S. Its first version was released in 1992 as a result of the cooperation between the FHWA and several State DOTs. In August, 1993, Pontis became a product of the AASHTO who released several versions;

the latest one is Pontis 3.4.1 released in September, 1998. Pontis is currently licensed through the AASHTO to 44 users: 37 State DOTs; 5 municipalities/other agencies; and 2 international users (AASHTO 1999). Pontis is a comprehensive BMS that supports the following (Thompson and Shepard 1994): (i) collecting bridge inventory and inspection data; (ii) formulating network-wide preservation and improvement policies for use in evaluating the needs of each bridge; and (iii) developing recommendations for an agency' s capital plan to derive the maximum benefit from its limited funds. Pontis employs a network optimization model and uses Markovian deterioration models to predict future conditions. For more detailed information about Pontis features, refer to Thompson et al. (1998).

BRIDGIT is a BMS developed by the National Cooperative Highway Research Program (NCHRP) in the U.S. It is very similar to Pontis in terms of its functions and capabilities. The primary difference between them lies in the optimization model. BRIDGIT adopted the bottom-up approach to optimization while Pontis uses the top-down approach (Lipkus 1994). The advantage of the former is that BRIDGIT can perform multi-year analysis and consider delaying actions on a particular bridge to a later date. BRIDGIT was first introduced in the fall, 1993.

Only a handful of States in the U.S. have opted to develop and implement their own BMS such as North Carolina and Indiana. In Canada, there is no dominant BMS that is used by all the provincial ministries of transportation, but most of these ministries of transportation have their own BMS's. The Ministry of Transportation of Ontario (MTO)

initiated its system, which includes 3,400 highway structures, in 1988 and updated it in 1999. This system consists of the Ontario Structural Inventory System (OSIS), which includes all design, construction, and technical data for Ontario's highway structures (bridges, culverts, retaining walls, etc.) and the Ontario Structure Inspection Management System (OSIMS), which includes condition ratings for all structural and non-structural elements (MTO 1991). The system makes use of decision trees to identify viable maintenance options and employs Markovian deterioration models to predict future bridge condition (Thompson 1998). Cost models are also employed to select optimum maintenance action on both project and network levels.

In Quebec, the Ministry of Transportation of Quebec (MTQ) initiated its BMS, which includes 9,500 highway structures, in 1996. The system is titled "Système de Gestion des Structures" (SGS) and it ranks bridges for eligibility to maintenance without using any deterioration models (MTQ 1997). More information about this system will be presented later in Chapter 7.

In western Canada, six municipalities developed a BMS for municipal sized inventories. This system consists of two database modules: static for inventory data, and dynamic for visual inspection data (Kriviak 1999). Analysis routines are also provided to compute sufficiency ratings of bridges, establish network-wide maintenance strategy options, and facilitate present value computations and the prediction of least-cost long-term maintenance.

2.3 BRIDGE DETERIORATION MODELS

Early versions of BMS's have used different methods for ranking network bridges. Ranking is a simple prioritization of bridges based on a formula that considers bridge condition, function, use, and importance. This ranking prioritizes bridges for eligibility to annual improvement funds i.e., bridges with the worst condition and highest importance have the highest priority in the treatment plan while bridges with better condition and lower importance are treated in a deferred plan. One of these methods is known as the Deficiency Point rating (Scherer and Glagola 1994). This rating prioritizes bridges based only on their physical characteristics such as material defects. Another method, called the Level-of-Service method, prioritizes bridges according to their overall condition rather than physical characteristics (Scherer and Glagola 1994). Using this method, certain condition ratings trigger the need for maintenance action. However, the Level-of-Service method does not include planning of future maintenance.

The FHWA defined a Sufficiency Index (SI) formula to provide a simple basis for ranking projects and screening the network to identify general MR&R needs (Hudson et al. 1987). SI is calculated based upon the NBI data items related to bridge structural condition, functional obsolescence, and importance for public use. The method of SI has been widely used by States DOT's. It has been extended by including decision trees that make MR&R action selections for each bridge according to a predefined set of decision criteria. These decision trees incorporate more information about each bridge before ranking, and consequently, generate better prioritization.

Other methods of prioritizing bridges have been introduced by provincial ministries of transportation in Canada. For example, the MTQ calculates "Indice Combiné d'une Structure" (ICS) based on structure functionality, condition, and seismic vulnerability (MTQ 1997). The higher the value of ICS, the higher the level of bridge functionality, the better the bridge condition, and the less its seismic vulnerability.

None of these ranking methods considers the bridge life cycle in prioritizing bridges and in selecting the most eligible bridges for maintenance. They also do not consider the long-term effects of the improvement actions in selecting the optimal maintenance strategy. Neglecting bridge life cycle and disregarding long-term effects of improvement actions result in misleading bridge rankings and consequently poor selections. Therefore, following any of the previously mentioned methods restricts the benefits obtained from using bridge management systems. Moreover, it may be inaccurate to call such systems management systems because they do not provide transportation agencies with an adequate tool for planning the future needs, estimating the condition of bridge networks, and making real cost-effective decisions.

In order to improve the situation, extensive research has been carried out since the late 1980's to construct deterioration models that predict the future bridge condition and, consequently, facilitate the financial analysis of different maintenance strategies and the determination of the optimal strategy. Below is a detailed literature review of the deterioration models used in BMS's and in other types of infrastructure management systems (IMS's). For clarity purposes, the intended meaning of the word "deterioration"

mentioned throughout this study stands for the decline in the facility condition resulting from normal operating conditions excluding any emergency cases like earthquakes, floods, fire, accidents, winds, etc.

The literature on IMS's is full of various approaches for modeling facility deterioration. The first approaches were developed for predicting pavement condition in pavement management systems (PMS's). The nature of the deterioration process in pavement is different from the nature of deterioration in bridges. This is due to the differences in construction material, structure configuration, and load type that consequently lead to differences in the patterns of resulting distresses. In addition, measures of bridge condition represent both bridge safety and serviceability, while measures of pavement condition represent pavement serviceability only. Despite the dissimilarities in the nature of deterioration between pavements and bridges, approaches used in developing pavement deterioration models for PMS's have been employed in developing bridge deterioration models for BMS's.

Existing bridge deterioration models can be categorized, as shown in Figure 2.2, into four main categories: Mechanistic Models, Deterministic Models, Stochastic Models, and Artificial Intelligence Models. Figure 2.2 also shows the sub-categories and the techniques used in each of the main categories. A detailed discussion on each category, sub-category, and technique is presented in the following subsections.

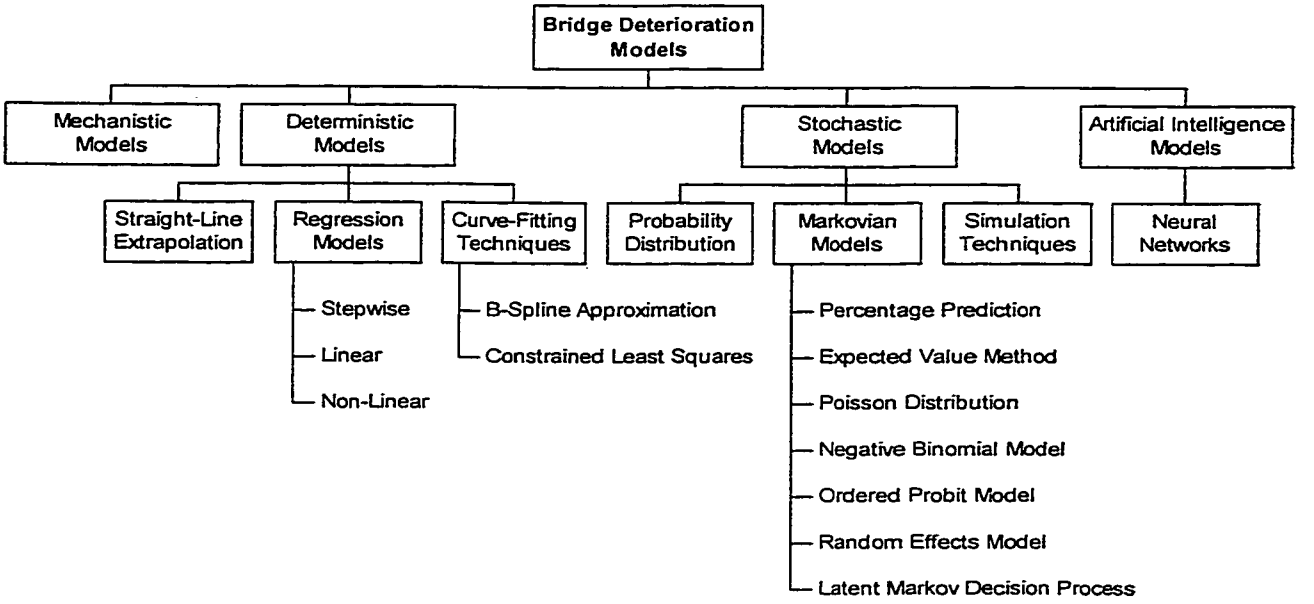


Figure 2.2: Categories of bridge deterioration models

2.3.1 MECHANISTIC MODELS

Mechanistic models are detailed models that describe specific deterioration mechanisms of particular bridge components. These models are usually effective for analysis at the project level, but they are difficult to develop for practical applications at the network level. A good example of such models is presented by Kayser and Nowak (1989). This mechanistic model is based on the observation that corrosion loss in steel superstructure follows an exponential law (Sobanjo 1991):

$$C = A\, t^{B}$$

Where,

C Average corrosion penetration in microns

t Time in years

A, B Parameters determined from observations.

From this model, predictions of bridge steel girder rating can be developed. Using this exponential law and making certain assumptions about the relative sensitivity of various portions of bridge girders to corrosion, models can be generated to estimate the remaining capacity for resisting moments, shear, and web buckling (Sobanjo 1991).

Another study on progressive deck deterioration was carried out by Cady and Weyers (1984). In this study, deicing-salt was found to be the most important contributor to the corrosion of reinforcing steel, which produces fracture planes and spalling of the deck surface. This corrosion is initiated when the chloride ion exceeds the so-called "threshold concentration" at the level of the top reinforcement. The corrosion must progress for a period of time before the corrosion products develop sufficient pressure to cause concrete to crack. Routine maintenance (e.g. patching) is a practical action as long as the area of spalling is limited. But, when the deteriorated area represents a high percentage of the deck area (e.g. 40%), routine maintenance becomes impractical and deck rehabilitation becomes necessary. The deck age at rehabilitation can be estimated as follows (Cady and Weyers 1984):

$$T_r = T_{cor} + T_{cra} + \left(\frac{Z - 0.5X}{R} \right)$$

Where,

T_r Deck age at rehabilitation in years.

T_{cor} Time to initiation of corrosion.

T_{cra} Time to cracking due to deicing-induced corrosion.

Z Percent of deck rehabilitation at overlaying.

X Percent of top reinforcing steel corroded due to subsidence cracking.

R Average deterioration rate per year.

The values of the previous parameters are calculated from the statistical analysis of a large number of bridge data.

2.3.2 DETERMINISTIC MODELS

Deterministic models are causal models that depend on a mathematical or statistical formulation for the relationship between the factors affecting bridge deterioration and a measure of bridge condition. These models are efficient for the analysis of networks with large population. The output of such models is always expressed by deterministic values (i.e. there are no probabilities involved) that represent the average of predicted conditions. Deterministic models can be categorized into: Straight-line Extrapolation, Regression Models, and Curve-Fitting Techniques. The details of each category are presented in the following subsections.

2.3.2.1 Straight-line Extrapolation

The simplest condition prediction is based on a straight-line extrapolation of the last two condition ratings. This method can be used in predicting the material condition rating (MCR) in bridges. The method assumes that traffic loading and previous maintenance levels will continue as in the past. This method requires that at least one condition measurement has been performed since construction, thereby providing two points: an initial bridge condition that can be assumed at the time of construction, and a second bridge condition determined at inspection time. The Straight-Line Extrapolation is used

because of its simplicity and practicality. It is also used when it is not known whether the rate of deterioration is likely to increase or decrease as shown in Figure 2.3 (Shahin 1994).

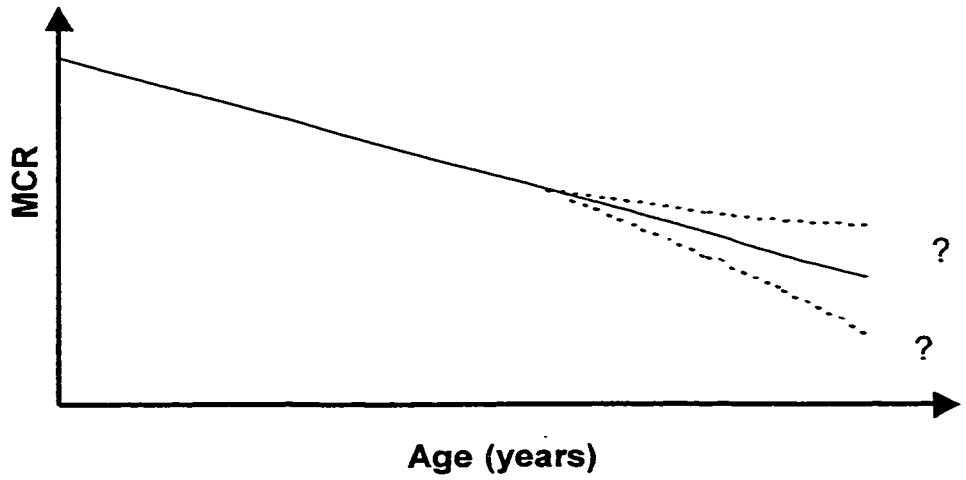


Figure 2.3 Straight-Line Extrapolation for condition prediction (Shahin 1994)

It should be noted that factors that affect the deterioration process such as climate, traffic, structure system, and bridge design could all be considered by segmenting bridge data into groups using combination of the different values of these factors. Then, a separate Straight-Line Extrapolation model is developed for each of these homogeneous groups to be used for predicting the future condition of bridges from this group only (Shahin 1994). Although this method of predicting deterioration is accurate enough for a short period of time (a few years), it is not accurate enough for long periods of time. Also, the Straight-Line Extrapolation method cannot be used to predict the rate of deterioration of a relatively new bridge or a bridge that has recently received major rehabilitation.

2.3.2.2 Regression Models

Regression models are used to establish an empirical relationship between two or more variables: one dependent (i.e. response) variable and one or more independent variables. Each variable is described in terms of its mean and variance (Shahin 1994). Several forms of regression models are used such as Stepwise Regression, Linear Regression, and Non-Linear Regression. These forms are described below.

Stepwise Regression

Stepwise Regression models can be used to predict material condition rating of bridge components. Several variables can be incorporated into these models, including bridge type, condition rating, nondestructive testing (NDT) results, bridge traffic, bridge age, and surrounding environment. These variables are grouped into categories, each of which is used to divide up the bridge data. The statistical significance of each division has to be tested by means of general correlation matrices (O'Brien III et al. 1983). These correlation matrices are used to select variables for further analysis. Stepwise Regression models were developed for predicting pavement condition index (PCI) for different pavement sections. These models were evaluated and determined to be unsatisfactory at the project level (O'Brien III et al. 1983). The resultant coefficient of determination (R^2) was very low and the residual standard error was high. This was partially attributed to the large number of estimation errors associated with these models. Another drawback to these models is that they are difficult to update to reflect additional condition data when they are collected.

Linear Regression

The simplest form for the relationship between a bridge condition and its age is the straight line. The linear relationship can be expressed by the following formula (Shahin 1994):

$$Y_i = \beta_0 + \beta_1 X_i + \varepsilon_i$$

Where,

Y_i	Condition rating of the bridge *i*.
X_i	Age of the bridge *i*.
ε_i	Prediction error of the bridge *i*.
β_0, β_1	Regression parameters

Naturally, there are many other explanatory variables that affect the bridge condition beside its age. The effect of these variables can be introduced by grouping bridges according to their type, class, location, and average daily traffic. This grouping technique guarantees the homogeneity among bridges of the same group. The disadvantage of this grouping technique is the limited amount of data per group, which reduces the model reliability.

Another way for considering the effect of these variables is using Multiple Linear Regression models. These models take the following form (Shahin 1994):

$$Y_i = \beta_0 + \beta_1 X1_i + \beta_2 X2_i + \beta_3 X3_i + \varepsilon_i$$

Where,

Y_i	Condition rating of the bridge *i*.
$X1_i, X2_i, X3_i$	Variables such as bridge age, type, location and so on.
ε_i	Prediction error of the bridge *i*.
$\beta_0, \beta_1, \beta_2, \beta_3$	Regression parameters

Non-Linear Regression

When the true relationship between the two variables (Y_i and X_i) is not linear, Linear Regression models do not provide sufficient accuracy and they may underestimate or overestimate the bridge condition at a given time. Accordingly, efforts for developing Non-Linear Regression models were needed to provide adequate prediction accuracy. A third-order polynomial model was found to be the best fit for the relationship of bridge component condition rating versus age. The polynomial model is expressed by the following formula (Jiang and Sinha 1989):

$$Y_i = \beta_0 + \beta_1 X_i + \beta_2 X_i^2 + \beta_3 X_i^3 + \varepsilon_i$$

Where,

Y_i	Condition rating of a bridge *i*.
X_i	Age of a bridge *i*.
ε_i	Prediction error of a bridge *i*.
$\beta_0, \beta_1, \beta_2, \beta_3$	Regression parameters

This equation indicates that the condition rating of a bridge depends on the bridge age only. The inclusion of other variables in this model increases the number of data groups and reduces the amount of data per group. This, consequently, reduces the reliability of the model and limits the data available for model verification. A specific study to include the effect of rehabilitation on the deterioration curve was established by Sanders and Zhang (1994). In this study, five factors that influence deterioration were considered by developing separate models for each combination of the five variables. The basic model used was an eight-parameter model as follows (Sanders and Zhang 1994):

$$Y(t) = (1-A)(1-B)(1-C)\alpha_1 e^{-t/\beta_1} + A\alpha_2 e^{-t-t_{r1}/\beta_2} + B\alpha_3 e^{-t-t_{r2}/\beta_3} + C\alpha_4 e^{-t-t_{r3}/\beta_4}$$

Where,

$Y(t)$	Bridge condition rating at age t.
t	Bridge age in years.
t_{r1}, t_{r2}, t_{r3}	Bridge ages in which major rehabilitation was conducted.
A, B, C	1 if the bridge is rehabilitated at times t_{r1}, t_{r2}, t_{r3} respectively and 0 any time else.
α, β	Regression coefficients to be estimated.

2.3.2.3 Curve-Fitting Techniques

Curve-fitting techniques are mathematical techniques that depend on constructing a polynomial that best fit bridge condition data to provide accurate condition predictions.

These techniques should be used after grouping bridge data into families and then subjecting them to a filtering procedure and an outlier's analysis (Shahin et al. 1987). Two Curve-Fitting techniques are available: B-Spline Approximation and Constrained Least Squares. Each of these techniques are described in the following subsections.

B-Spline Approximation

To address the problems resulting from using a polynomial curve fitting, a normalized B-spline function was chosen to approximate the MCR/age data. The spline assumes a shape that minimizes its potential energy (Shahin et al. 1987). This shape was found to be a cubic polynomial between each pair of selected points (i.e. knots). Adjacent polynomials join continuously with continuous first and second derivatives.

An important consideration in using this technique is the choice of knots for these polynomials. Sensible choices of the number and the position of the interior knots may often be estimated by examining the shape of the desired curve, but generally this is a process that requires advanced engineering judgement (Shahin et al. 1987). Failures in this selection process can result in functions that are not strictly decreasing. To develop such a deterioration model, age data has to be averaged on a n-year basis that produces smoother curves and reduces the occurrence of positive slopes. There are two causes for having a positive slope as shown in Figure 2.4. One is the presence of more than one group of the bridge data. Another cause is the existence of a bridge that has an unexpectedly high condition at some ages (e.g. point A).

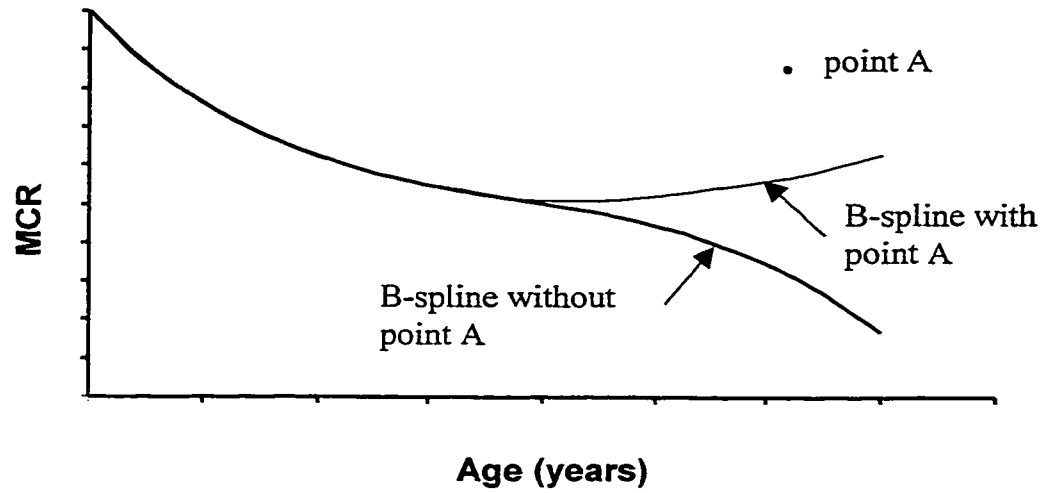


Figure 2.4 B-Spline Approximation for modeling deterioration (Shahin et al. 1987)

Constrained Least Squares

The Constrained Least-Squares Estimation approximates given observations using polynomial curve fitting. This polynomial is mathematically constrained by the requirement that the first derivative of the curve at any age is kept negative or zero (Shahin et al. 1987). This ensures that condition values do not increase with age. The following constrained linear least squares function was adopted:

$$\text{Minimize} \quad \sum_{k=1}^{N} [Y_k - P(x_k)]^2$$

$$\text{Subject to} : P(0) = 100 \text{ and } P'(x_j) \leq 0 \text{ for } x_j \text{ between } 0 \text{ and max. age}$$

Where,

Y_k Condition of bridge k

x_k Age of bridge k

$P(x)$ Bridge condition polynomial function of x

This function minimizes the total difference between bridge condition data points Y_k and the developed polynomial $P(x)$ where x is the bridge age. Using this technique to model bridge deterioration has several advantages (Shahin et al. 1987). Constrained Least Squares curves will never exhibit a positive slope (i.e. MCR increases with age). Also, it is simpler and more accurate than the B-Spline Approximation whenever the data file contains errors.

2.3.2.4 Discussion of Deterministic Models

Transportation agencies that have recently started developing a bridge management system do not have a recorded history for the conditions of their bridges. Agencies that have recently updated their inspection system or have significantly changed their condition rating scale do not also have a consistent history for the conditions of their bridges. All these agencies do not even have two successive and consistent condition ratings for the same bridge component. Under these limitations, transportation agencies are obligated to model bridge components deterioration deterministically as a function of component age and other explanatory variables (if available). But, these deterministic models have the following shortcomings:

1. Neglecting the uncertainty and randomness due to the existence of unobserved explanatory variables, presence of measurement error, and stochasticity of the deterioration process (Madanat and Ibrahim 1995).
2. Overlooking the current bridge condition when predicting the future condition, which sometimes results in having a predicted condition higher than the current one although no maintenance actions were considered (Sanders and Zhang 1994).

3. Segmenting bridges into groups of homogeneous explanatory variables, which results in a limited amount of data points per group and may, consequently, reduce model reliability (Shahin 1994).
4. Disregarding the interactive effects between the deterioration mechanisms of different bridge components, such as bridge deck, deck joints, and bearings (Sianipar and Adams 1997).
5. Difficulty of model updating when new bridge data are obtained or different rating systems are used.

2.3.3 STOCHASTIC MODELS

Stochastic models have made great contributions to the field of modeling infrastructure deterioration because the nature of the deterioration process is full of uncertainties and randomness. Stochastic models can be categorized into: Probability Distribution, Markovian Models, and Simulation Techniques. Each of these categories is described below in detail.

2.3.3.1 Probability Distribution

An infrastructure facility condition measure, such as material condition rating (MCR) for bridges, can be treated as a random variable with probabilities associated with its value. A probability distribution describes probabilities associated with all the values of a random variable. For example, if the random variable is the MCR, then its probability distribution can be described by its cumulative distribution function (CDF) as shown in Figure 2.5 (Shahin 1994). The vertical axis in this figure is the probability of MCR

(P_{MCR}) being less than or equal to a given value. This figure can be drawn at different time points in the life of a bridge. It can be presented also as probability versus time for a selected MCR value in what it is known as the "survivor curve". The probability drops off with time from a value of 1.0 down to 0.0 and it expresses the percentage of bridge that remains in service with a MCR greater than a selected value.

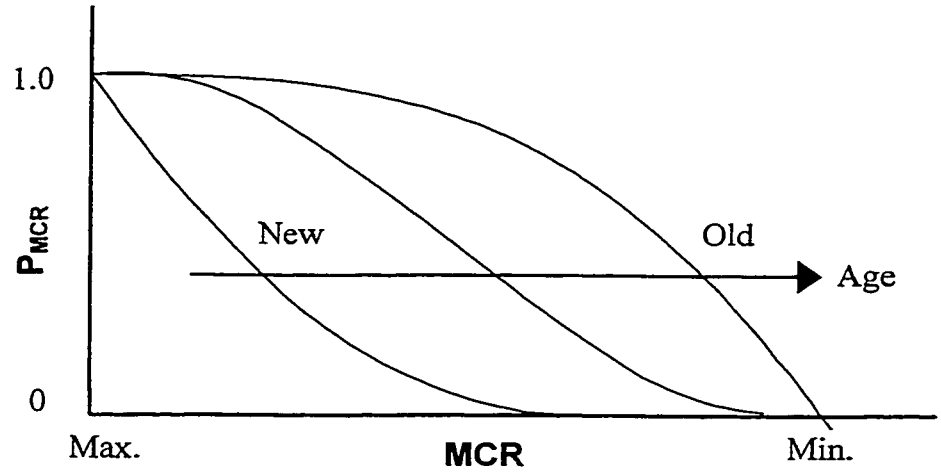


Figure 2.5: Cumulative distribution function of MCR (Shahin 1994)

The use of Probability Distribution in predicting bridge condition requires the knowledge of the distribution law for the variable being predicted. This technique is particularly useful for individual distress prediction.

2.3.3.2 Markovian Models

One of the most popular stochastic techniques obtained from operations research is the Markov Decision Process (MDP). This process was used extensively in developing stochastic deterioration models for different infrastructure facilities (e.g. pavements, bridges, pipes, buildings, etc.). Markovian deterioration models are based on the concept

of probabilistic cumulative damage, which sums up the facility deterioration over multiple transition time intervals. Markovian models assume discontinuous MDP that defines distinctive states of facility condition rating and discrete transition time intervals. Probabilities of transition from one condition state to another are represented in a matrix (n x n) that is called the transition probability matrix P where n is the number of condition states. Each element in this matrix $p_{i,j}$ represents the probability that the bridge will make a transition from state i to state j during a given period of time. If the present or the initial state of the bridges is known, the future condition can be predicted through multiplication of the initial state vector and the transition probability matrix. The future condition after any time T expressed by the future-state vector P_T can be obtained by the multiplication of the initial-state vector P_0 and the transition probability matrix P raised to the power T as follows (Collins 1972):

$$P_T = P_0 * P^T$$

$$\text{Where, } P = \begin{vmatrix} P_{1.1} & P_{1.2} & \ldots\ldots & P_{1.n} \\ P_{2.1} & P_{2.2} & \ldots\ldots & P_{2.n} \\ . & . & \ldots\ldots & . \\ . & . & \ldots\ldots & . \\ P_{n.1} & P_{n.2} & \ldots\ldots & P_{n.n} \end{vmatrix}$$

Markovian deterioration models also assume that the future condition state depends only on the present condition state, but not on the past condition states. This state independence of Markovian models is known as a first-order MDP or the Markov Property. Moreover, Markovian models assume constant bridge population and stationary transition probabilities throughout the prediction period. Verifying the applicability of

Markovian assumptions and determining reliable transition probabilities are the most difficult problems in developing Markovian deterioration models (Madanat et al. 1995). Several research efforts have focused on solving the problem of determining transition probabilities using different techniques, while few efforts have been devoted to verifying Markovian assumptions. Below is a discussion on each of these efforts listed in a chronological order.

Percentage Prediction Estimation

The probability of transition in bridge condition from one state to another can be estimated easily from a bridge inspection database using the following equations:

$$P_{i,j} = n_{i,j} / n_i$$

$$n_i = \Sigma n_{i,j}$$

Where,

$n_{i,j}$ Number of transitions from state i to state j within a given time period.

n_i The total number of bridges in state i before the transition.

Consequently, the transition matrix can be determined and the prediction can be performed. But, the resulting transition matrix is independent of the bridge age; in other words, it is not homogeneous with respect to the bridge age. Therefore, the Percentage Prediction Estimation is not recommended for developing bridge performance curves in spite of its simplicity (Jiang et al. 1988).

Expected Value Method

The Expected Value Method is the most commonly used approach for estimating transition probabilities. It consists of the following three steps (Madanat et al. 1995):

1- Bridges are classified into groups where each group consists of bridges having similar attributes.

2- For each group a deterioration model with a condition rating Y, as dependent variable, and an age t, as independent variable is estimated. The mathematical expression is:

$$Y_n = \beta_0 + \beta_1 t_n + \varepsilon_n$$

Where,

n Bridge index.

β_0, β_1 Parameters to be estimated by linear regression.

ε Random error.

3- A transition probability matrix is estimated for each group by minimizing the distance between the expected value of the bridge condition rating as predicted by a regression model and the expected value derived from the Markovian model. The mathematical representation of the non-linear objective function for the minimization process is as follows:

$$\text{Minimize} \quad \sum_{n=1}^{N} \left| Y_n - E(t_n, P) \right|$$

$$\text{Subject to} \quad : \; 0 \le p_{ij} \le 1 \qquad i, j = 1,2,\ldots\ldots k \qquad \sum_{i=1}^{k} p_{ij} = 1$$

Where,

N Total number of bridges.

p_{ij} Transition probability from state i to state j.

P Transition probability matrix.

$E(t_n,P)$ Expected condition of bridge n at age t using transition matrix P.

k Maximum value for the bridge condition rating.

Replacing the linear regression model with a third degree non-linear regression model can refine the accuracy of this estimation process (Butt et al. 1987).

Poisson's Regression and Negative Binomial Models

Poisson's Regression Model is used to construct a discrete incremental deterioration model where the dependent variable is the number of drops in a bridge condition state during one inspection period (Madanat and Ibrahim 1995). For a bridge n with condition state i, the dependent variable Z_{in} can be any integer value j greater than or equal to 0, and less than or equal to i.

In MDP, the amount of time a bridge spends in each state is exponentially distributed, and independent of the amount of time the bridge spends in its previous state. Consequently, the number of drops in its condition state within an inspection period follows a Poisson distribution. The Poisson probability mass function is as follows (Madanat and Ibrahim 1995):

$$p(Z_{in} = j | X_n, i) = \frac{e^{-\lambda_{in}} \lambda_{in}}{j!}$$

$$\lambda_{in} = e^{\beta X_n}$$

Where,

β Vector of parameters to be estimated

X_n Vector of explanatory variables such as age, traffic, material, wearing surface, environmental factors, etc. for bridge n

Maximum Likelihood Estimation (MLE) is used to estimate the value of the parameters vector β. Once these parameters are estimated, they can be used to compute transition probabilities as a function of time. The Poisson's Regression Model has major advantages over the commonly used Expected Value Method (Madanat and Ibrahim 1995). First, it explicitly links deterioration to relevant explanatory variables and, consequently, does not need the segmentation required by the expected value to have a homogeneous population. Second, it recognizes the discrete nature of the dependent variable while the linear regression assumes continuity. One of the fundamental shortcomings of the Poisson distribution is that it assumes the variance of the random variable to be equal to the mean. This situation is not common in real world problems to encounter, and to overcome this shortcoming the Poisson's Regression Model can be replaced with a Negative Binomial Model that allows the mean of the dependent variable to be different from the variance. This was done by redefining λ_{in} of the Poisson distribution to be a random variable λ^*_{in} that can be written as follows (Madanat and Ibrahim 1995):

$$\lambda_{in}^{*} = e^{(\beta X_n + \varepsilon_n)}$$

Where,

β, X_n The same as in the Poisson distribution.

ε_n Random error term representing omitted explanatory variables.

Ordered Probit Model

The Ordered Probit Model assumes the existence of an underlying continuous unobservable random variable and, therefore, allows for capturing the latent nature of infrastructure performance (Madanat et al. 1995). This model is used to construct an incremental discrete deterioration model for bridge decks. For a given condition state i, of a bridge n, the dependent variable Z_{in} is the difference between the condition states in two consecutive inspections. This difference is an indicator of the change in the latent performance U_{in}. This change can be presented as a function of explanatory variables X_n as follows (Madanat et al. 1995).

$$\log(U_{in}) = \beta X_n + \varepsilon_{in}$$

Where,

β Vector of parameters to be estimated.

In order to estimate this vector of parameters, thresholds are used to map the continuous values of U_{in} into discrete values of Z_{in} (Madanat et al. 1995). Then, the MLE is used to estimate the value of this vector and these thresholds simultaneously. Then, the transition probability from state *i* to state *i-j*, where *j* is the number of drops in the bridge condition, is equal to the probability that the change in the condition state Z_{in} is equal to *j*. This probability is equal to the area under the density curve $f(\varepsilon_{in})$ between the corresponding U_{in} values.

Random-Effects Model

Incremental models, which were used previously to compute infrastructure transition probabilities, predict changes in conditions over time as a function of a set of explanatory variables. These models have typically been developed using data sets of in-service facilities, but have not accounted for the presence of heterogeneity, which is likely to exist in such data (Madanat et al. 1997). Heterogeneity refers to the presence of persistent facility-specific but unobserved factors, such as construction quality, and leads to biased model estimates. Another limitation of such models is that they are based on a Markovian representation of facility deterioration, which assumes that facility deterioration is independent of history. This assumption may be unrealistic for some types of facilities in which early stress initiation leads to accelerated deterioration later.

The Random–Effects Model is used to capture heterogeneity and to empirically test for the presence of true state dependence. A model of bridge-deck deterioration that captures possible heterogeneity in the data is defined as follows (Madanat et al. 1997):

$$Y_{it} = bX_{it} + u_i + v_{it} \qquad i = 1,....,N \quad t = 1,....,T_i$$

Where,

Y_{it} Change in latent deterioration.

b Vector of parameters to be estimated.

X_{it} Vector of explanatory variables for facility i in time period t.

u_i Individual bridge deck random effect.

v_{it} Random error term.

N Number of bridge decks in the sample.

T_i Periods of observation for bridge deck i.

The presence of the true state dependence is investigated by incorporating a lagged dependent variable as one of the explanatory variables in the model. By examining the significance of these explanatory variables, it was noted that most of these variables have become insignificant (Madanat et al. 1997). Also, the age variable has become insignificant, which is inconvenient because it implies that it is not possible to empirically distinguish between the effects of the state dependence and the age variable.

Latent Markov Decision Process

Deterioration models that are based on the MDP do not take into account the uncertainty in the measurement of facility condition. This uncertainty affects MR&R decisions because a measurement error will lead to the selection of a wrong activity. A methodology called Latent Markov Decision Process (LMDP) explicitly recognizes the

presence of random errors in the measurements of a facility condition. In this methodology, two major inputs must be described: measurement probabilities and transition probabilities. Measurement probabilities can be obtained, empirically, using measurement error models such as (Ben-Akiva et al. 1993):

$$d_r = \alpha_r + \beta_r d + \varepsilon_r$$

Where:

d	True condition of an infrastructure facility.
d_r	Measured condition of this facility by a technology r.
α_r, β_r	Addition and multiplication measurement biases of technology r.
ε_r	Random error of measurement by technology r.

Given a sample of facility condition using different technologies, the parameters of such a model can be statistically estimated. The estimated variance of the measurement error is then used to compute the measurement probabilities. Transition probabilities can be estimated by any of the methods described earlier. The LMDP methodology treats facility performance as a latent variable (Ben-Akiva et al. 1993). This variable S is linked to a vector of explanatory variables X and a vector of maintenance actions A through a deterioration model. Moreover, it is linked to a vector of condition indicators D through a measurement model. This methodology forms a system of equations to be solved simultaneously, thereby producing a much better fit to data than traditional deterioration models. Figure 2.6 shows a schematic representation of LMDP.

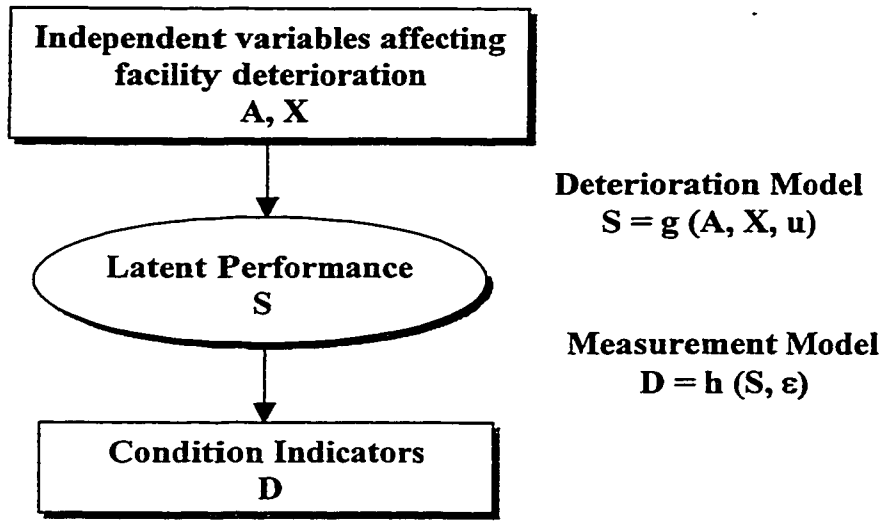


Figure 2.6: Integrated model of latent facility performance and condition measurement (Ben-Akiva et al. 1993)

2.3.3.3 Simulation Techniques

Since bridge deterioration is a complex process in terms of transition times between various states or conditions, the stochastic analysis of the system can also be achieved by means of a simulation model of the bridge performance over time. If statistics on transition times are available, the entire deterioration process can be simulated. The output of this simulation will be a probabilistic deterioration profile in terms of time taken by the bridge condition to change from one rating to another. It should be noted that the idea of developing a simulation model for bridge deterioration has not been tested practically (Sobanjo 1991).

2.3.3.4 Discussion of Stochastic Models

Stochastic Models have been used by experienced transportation agencies that have a relatively large bridge population with a recorded history of bridge conditions. These techniques have treated the two basic problems of Deterministic Models by capturing the

uncertainty of the deterioration process and considering the current bridge condition in predicting the future one. The main drawback of the Probability Distribution and Simulation techniques is that they have not been applied or tested practically. Although Markovian models have been employed by many advanced BMS's, such as Pontis and BRIDGIT, and have achieved great advances in modeling bridge deterioration, they have some assumptions and limitations that can be summarized as follows:

1. Markovian models assume the presence of state independence that means past conditions have no effect on the predicted ones (Madanat et al. 1997).
2. Markovian models assume discrete transition time intervals, constant bridge population, and stationary transition probabilities (Collins 1972).
3. Markovian models are incapable to consider the effect of major maintenance treatments on the deterioration process (DeStefano and Grivas 1998).
4. It is quite difficult to consider the interactive effects among deterioration mechanisms of different bridge components (Cesare et al. 1992).
5. Transition probabilities are estimated based on subjective engineering judgement and require frequent update using the Bayesian approach when new data are obtained (Tokdemir et al. 2000).

2.3.4 ARTIFICIAL INTELLIGENCE MODELS

The area of artificial intelligence (AI) comprises several different techniques that have been utilized in a variety of applications during the last few decades. Artificial Neural networks (ANN) are one of these techniques that has been recognized as a powerful tool in solving engineering problems. Pattern recognition, classification, diagnosis,

estimation, optimization, and prediction are the most successful areas of applications. An ANN is an information-processing system that has performance characteristics in common with biological neurons (Hua 1996). It emulates the learning capabilities of the human brain. The performance of ANN depends mainly on training cases; this performance is evaluated by using a set of unseen cases.

The feasibility of using ANN in modeling bridge deterioration has been investigated by Sobanjo (1997). A multi-layer Perceptron type of ANN was utilized to relate the age of bridge superstructure (in years) to its condition rating (in a numeric scale from 1 to 9). The network configuration used in this study had one input layer, two hidden layers and one output layer as shown in Figure 2.7. The inspection records for 50 bridge superstructures were utilized in training and testing the network; 75% of the data were used for training while the remaining were used for testing. A reasonable percentage (79%) of the predicted values matched the actual values with allowing a prediction error up to 15%. Using the same data fed into the input layer, a simple regression model was formulated between the condition rating and age of the bridge superstructure. The relationship obtained had a coefficient of determination, R^2=0.84.

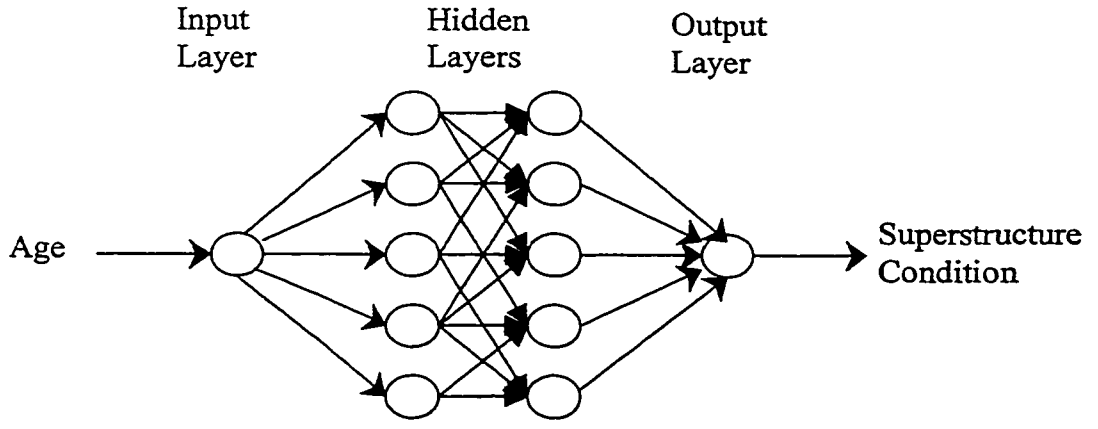


Figure 2.7: Multi-layer Perceptron neural network (Sobanjo 1997)

In Sobanjo's small-scale investigation, it was concluded that the ANN could be applied to model the deterioration of highway bridges. Nevertheless, a great effort is still needed to incorporate several deterioration factors other than the superstructure age and a large-scale investigation should be carried out using more data records in order to confirm the contribution of ANN to modeling bridge deterioration. A more detailed investigation was carried out by Tokdemir et al. (2000) to predict the bridge sufficiency index (SI) (i.e. ranging from 0 to 100) using age, traffic, geometrical, and structural attributes as explanatory variables. Testing the performance of the developed ANN's resulted in an average percentage of correct solutions equal to 33.5% and 62.5% with a prediction error equal to 3% and 6% respectively. These investigations revealed that ANN's have solved the problem of choosing the degree of the regression model that fits data points in the best way. On the other hand, it showed that ANN still has the other problems of Deterministic Models in addition to the following problems:

1- The determination of an efficient ANN architecture is carried out in an ad hoc manner and does not follow clear rules (Boussabina 1996; Hua 1996).

2- The use of ANN is limited to input and output variables that have numeric data or converted symbolic data. This conversion may lead to losing some information that was contained in the original representation (Arditi and Tokdemir 1999).

2.4 RESEARCH OBJECTIVES

The main objective of this research is to develop a realistic, accurate, and generic bridge deterioration model that addresses the deficiencies of the models described earlier and eliminates their limitations. Deterministic and stochastic models are the most

commonly used models in state-of-the-art BMS's. This is due to the fact that they are well-established models (i.e. tested by many transportation agencies) and practically feasible for large-sized bridge networks. The common limitations of these models can be summarized as follows:

1. Overlooking the effect of past bridge conditions on the predicted ones.
2. Neglecting the effect of major maintenance treatments on the deterioration process.
3. Using the segmentation technique to account for multiple explanatory variables.
4. Disregarding the interaction among different bridge components.
5. Difficulty of model updating when new data are available or different rating systems are used.

The prospective deterioration model is intended to be:

1. Practical:
   - Workable with bridge data that are available in transportation agencies.
   - Compatible with other BMS modules (e.g. optimization and cost modules).
   - Updateable whenever new data are available.
   - Capable of performing "what-if" analysis for different maintenance decisions.
2. Accurate:
   - Flexible in including the main factors that affect the deterioration process.
   - Capable of accounting for the effect of improvement actions done in the past.
   - Efficient in considering the effect of previous conditions on predicted ones.
   - Able to recognize the interaction among different bridge components.

3. Versatile:
    - Workable with different condition rating systems.
    - Able to consider bridges with variable inspection period.
    - Applicable for different bridge components.
    - Valid for various bridge types and construction materials.

2.5 SCOPE OF THE STUDY

This research is concerned with modeling the deterioration of short-to-medium-span highway bridges only regardless of their construction material. Bridges are defined as crossing structures that have a total length equal to or greater than a particular value. This value varies from a transportation agency to another. In the United States, the minimum bridge length is 6 m (20 feet), while it is 4.5 m (15 feet) in Ontario and 3 m (10 feet) in Quebec. In this research, bridges with a minimum total length of 3 m are considered. This is because data about Quebec bridges will be used in developing and testing the prospective deterioration model.

Bridges are also classified according to the span length into short-span, medium-span, and long-span bridges. Medium span bridges are defined as those whose longest span ranges between 60-120 meters (200-400 feet) (Sen et al. 1999). Short and long span bridges are defined as those whose longest span is, respectively, less or more than this previous range. Long-span bridges are excluded from this study because they have special features regarding their deterioration, inspection, and maintenance. These features can be found in Honshu-Shikoku (1994).

CHAPTER 3

THE PROPOSED METHODOLOGY

3.1 INTRODUCTION

During the last decades, the area of artificial intelligence (AI) has provided several approaches that have been broadly used for solving numerous problems in engineering and management domains. Rule-based expert systems (ES) and artificial neural networks (ANN) are the most popular approaches used by researchers in developing AI applications for both academia and industry. ES's have some restrictions in their performance and expandability (Roddis and Bocox 1997): they fail to give any solution to a problem whenever no rule is available to handle this problem; they cover just a portion of the problem domain, and it is difficult to update their rule bases when new problems are encountered; and they require knowledge acquisition from domain experts, which is an inconvenient and time-consuming process. Also, ANN's have some limitations in terms of their behavior and applicability (Arditi and Tokdemir 1999): they deal only with problems whose inputs and outputs are numeric or converted symbolic attributes, this may lead to losing the information contained in the original representation; they cannot handle efficiently problems that have incomplete data or variable data structure; and they have to be re-trained and re-tested when new data are available, which is a cumbersome and time-consuming task.

Case-based reasoning (CBR) is a relatively new AI approach that overcomes most of the drawbacks of ES's and ANN's by reusing the specific knowledge encapsulated in

previously experienced problems when solving similar ones. This approach is proposed in this research to solve the problem of modeling bridge deterioration and to satisfy the required objectives presented earlier in Chapter 2.

This chapter starts with a background on CBR approach and how this approach can be used in solving the research problem. Then, the development cycle of the proposed CBR system is introduced. This is followed by the definition of the system requirements, its expected functionality, and the resources required for the system development.

3.2 CASE-BASED REASONING

3.2.1 BACKGROUND

According to Maher et al. (1995), CBR is defined as "an approach to problem solving that makes use of a case library of previously solved problems when solving a new one, where a case library is a collection of stored cases that represent problem solving episodes". CBR systems are computerized tools that imitate the analogical reasoning of human brains in problem solving (Rivard et al. 1998). CBR systems propose solutions to problems the same way human experts find solutions for new problems by relying on their own experience with similar problems that were encountered before. The main source of knowledge in CBR systems is the cases that can be reused even if they are partially matching the problem in hand. This knowledge can be easily extended by adding new cases and without facing the problems of knowledge acquisition or rule coverage as in ES's (Roddis and Bocox 1997). Also, CBR systems can deal efficiently with both numeric and symbolic data, and can handle effectively cases that have incomplete data or

variable data structures (Arditi and Tokdemir 1999). Furthermore, CBR systems have powerful learning capabilities that do not require time-consuming training and testing operations (Yang and Yau 1996). A detailed literature on CBR systems and their evolution can be found in Watson and Marir (1994).

Case-based reasoning systems, in general, have four main aspects (see Figure 3.1):

1. **Case representation** organizes the information that describes case contents (i.e. the problem, the solution, and the outcomes) in the system case library (Kolodner 1993). AI provides a wide range of representational paradigms (e.g. frames, semantic networks, attribute-value pairs, object-oriented representation, etc.) that can be used to represent the contents of a case in a form that the computer can read (Maher et al. 1995).
2. **Case accumulation** is responsible for storing new cases into the case library to support CBR learning capabilities. Learning in CBR systems happens by the addition of specific cases, unlike traditional symbolic and neural network approaches, which learn by generalization and discard the cases on which the generalization is based (Leake 1996).
3. **Case retrieval** queries the case library for the most relevant case(s) to the current problem in order to reuse them. The retrieval process in CBR differs from that of Information Retrieval (IR) systems and standard database (DB) systems, which leave the problem of how to formulate the right query largely to the user. In CBR systems, the system itself is often designed to determine appropriate retrieval cues. DB systems are designed to do exact matching between queries and stored

information, while the goal of CBR is to retrieve the "most similar" case or a set of most similar cases which are called proposed solutions (Leake 1996).

4. **Case adaptation** revises the retrieved case(s) to fit the current problem context. The difficulty of the case adaptation process results from its dependency on the domain knowledge which is not always available in cases (Watson 1997)

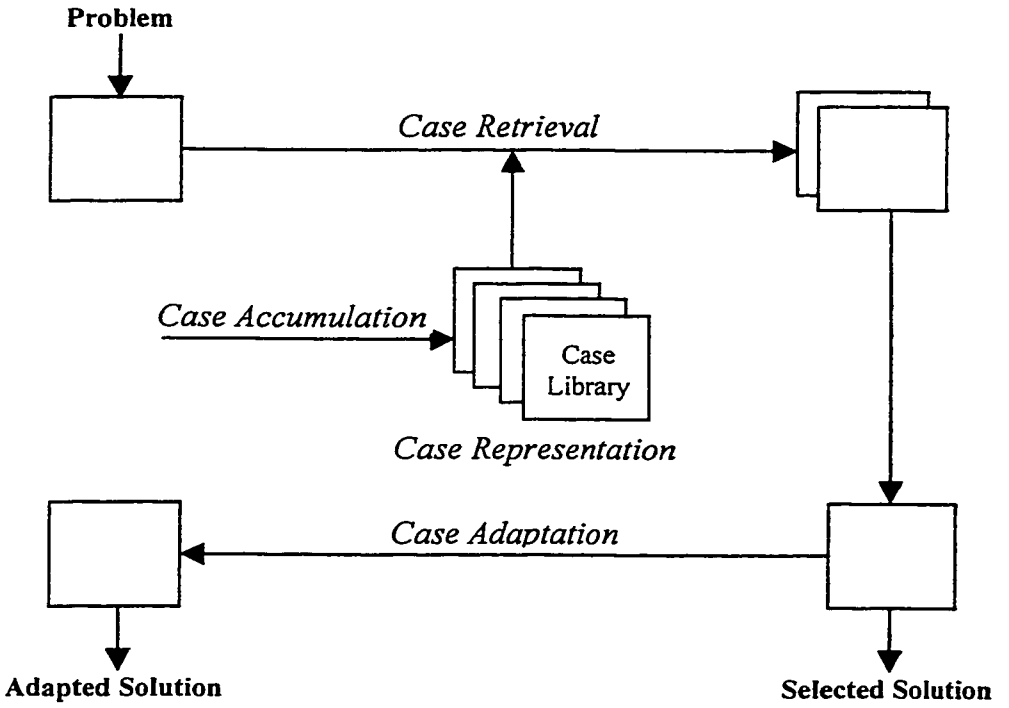


Figure 3.1: CBR process and its main aspects

Figure 3.1 depicts how CBR works to solve a given problem. First, CBR searches the case library for stored problems that are similar to the given one and retrieves them. Second, CBR assesses the similarity of the retrieved problems and selects the solution of the best-matched problem. Third, CBR adapts this solution for the differences between the given problem and the retrieved problem. The adapted solution is, then, presented to the user as the final solution of the given problem. Fourth, CBR updates the case library by accumulating new problems that encapsulate human experience in their solutions.

CBR systems are useful for a wide variety of tasks. These tasks can be categorized into problem solving tasks, and interpretation tasks as shown in Figure 3.2 (Kolodner 1993). Problem solving CBR systems use the experience stored in prior cases to assist users in solving problems that require the achievement of multiple objectives or the satisfaction of multiple constraints. Problem solving tasks include planning, design, and diagnosis. Interpretative CBR systems are useful when no computational method is available to evaluate the solution or there are so many unknowns and the available knowledge is insufficient to provide a solution. Interpretation tasks include classification, adversarial reasoning, and projection.

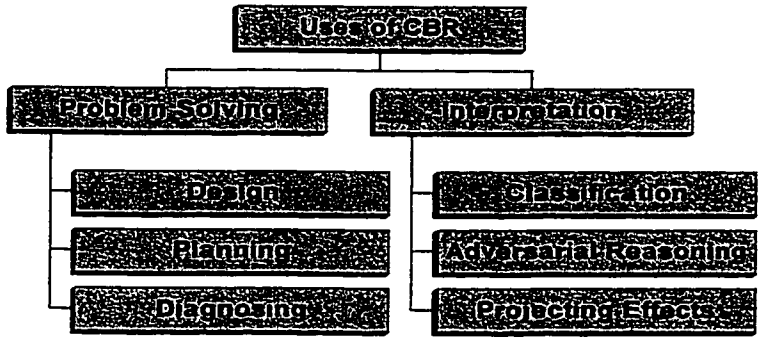


Figure 3.2: Categories of CBR uses (Kolodner 1993)

Modeling bridge deterioration can be categorized as an interpretative task because predicting the future condition of a bridge consists in projecting effects and consequences of maintenance decisions that have been taken in the past for that bridge. The closest example in the literature of CBR systems that have been used in projecting effects of a decision is Battle Planner (Goodman 1989). Battle Planner is a CBR system that plays a coaching role for novice battle planners. The user describes a battle situation to the system and tells the system about his decision. The system then retrieves similar situations that can be used in projecting the consequences of this decision.

3.2.2 CBR FOR MODELING BRIDGE DETERIORATION

The basic idea of using CBR in modeling bridge deterioration assumes the similarity in the performance of different bridges that have similar physical features (e.g. material, structural systems, cross section, span, etc.), environmental and operating conditions (e.g. traffic, region, highway class, etc.), and inspection and maintenance history. This assumption will be justified in this research throughout the development of a "proof of concept" CBR system. Although the probability of having bridges that are similar in their physical features, environmental and operating conditions, and inspection and maintenance records is not high, carrying out the matching process at the level of bridge components increases the probability of finding similar components. Furthermore, increasing the number of bridges/bridge components in the BMS databases increases this probability and improves the system performance.

To illustrate the idea of using CBR in modeling bridge deterioration, consider 2 bridges: one is the query bridge that has an age Aq and inspection and maintenance records for a period Pq measured from now (time = 0); the other is the case bridge that has an age Ac and inspection and maintenance records for a period Pc measured from now. The periods Pq and Pc include the recent inspection and maintenance records that are the most important records for the matching process. Figures 3.3 and 3.4 show two examples of these query and case bridges with different values for their ages and inspection and maintenance periods.

In order to predict the future condition of the query bridge in the coming *Y* years using the condition records of the case bridge, the following requirements have to be satisfied:

1. The query bridge matches the case bridge in all available parameters that affect the deterioration process such as material, structural system, geometry, traffic, highway class, environment, etc.
2. There is an age, called prediction age *Ap*, at which the condition of the case bridge matches the current condition of the query bridge and the conditions records of the case bridges are available for at least *Y* years after *Ap*.
3. Inspection and maintenance records of the case bridge during the period from (*Ac-Pc*) to *Ap* match the corresponding records of the query bridge during the period from (*Aq-Pq*) to the present. When periods are different, matching occurs over the smaller period.

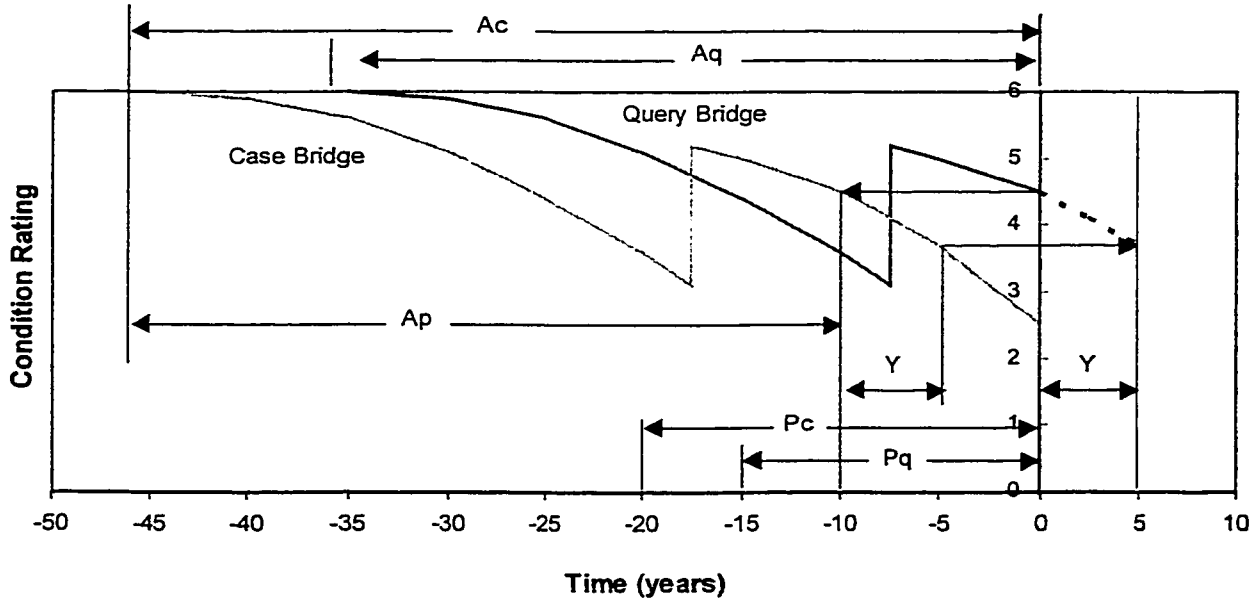


Figure 3.3: Example for a case bridge older than the query bridge.

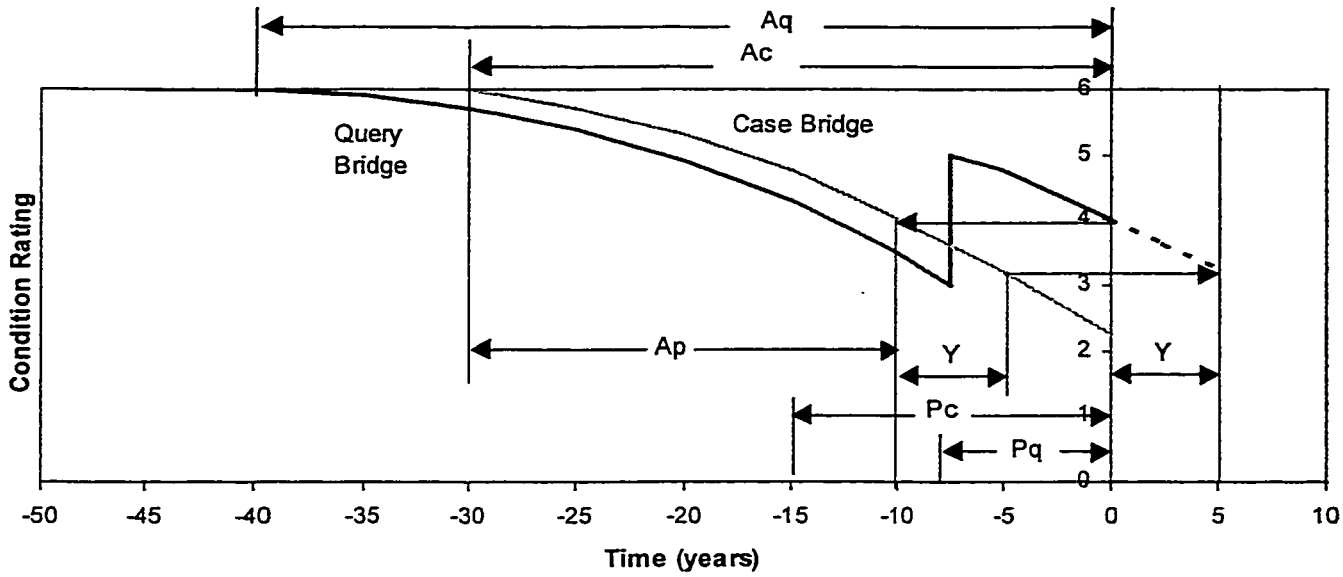


Figure 3.4: Example for a query bridge older than the case bridge

These three requirements are considered satisfied even if the matching is partial. However, the better the matching between the query bridge and the case bridge, the better the prediction. If the previous requirements are not satisfied because of the length of the prediction period Y, this period can be broken down into shorter periods, each of which provides a short-term prediction. The satisfaction of the first requirement, which is matching deterioration factors, is not as important as the second and third requirements, which are matching current and past inspection and maintenance records. This is because current and past bridge conditions already capture the effect of the deterioration factors in their values. However, in the situation where these records are unavailable, deterioration factors become very important in predicting the future condition.

Figure 3.3 shows an example where it is required to predict the bridge condition during the next 5 years ($Y = 5$) for a query bridge that is 35 years old ($Aq = 35$) and has inspection and maintenance records for a 15 year period ($Pq = 15$). A case bridge that is

45 years old ($Ac = 45$) and has inspection and maintenance records for a 20 year period ($Pc = 20$) was retrieved to predict the future conditions. This case bridge satisfies the second requirement: at the age of 35 years ($Ap = 35$), which is greater than 25 ($Ac - Pc = 45 - 20$) and less than 40 ($Ac - Y = 45 - 5$), this bridge had a condition equal to the current condition of the query bridge (condition rating = 4.5). Also, inspection and maintenance records of the case bridge during the period from 25 ($Ac - Pc$) to 35 (Ap) years old matches those of the query bridge during the period from 25 ($Aq - Pq$) to 35 (now) years old. Therefore, the condition of the case bridge at age 40 years old (condition rating = 3.75) can be used as a prediction for the future condition of the query bridge after 5 years from now, assuming that the two cases match in their deterioration factors.

Figure 3.4 shows an example where it is required to predict the bridge condition during the next 5 years ($Y = 5$) for a query bridge that is 40 years old ($Aq = 40$) and has inspection and maintenance records for an 8 year period ($Pq = 8$). A newer case bridge that is 30 years old ($Ac = 30$) and has inspection and maintenance records for a 15 year period ($Pc = 15$) was retrieved to predict the future conditions. This case bridge satisfies the second requirement: at the age of 20 years ($Ap = 20$), which is greater than 15 ($Ac - Pc = 30 - 15$) and less than 25 ($Ac - Y = 30 - 5$), this bridge had a condition equal to the current condition of the query bridge (condition rating = 4). Also, inspection and maintenance records of the case bridge during the period from 15 ($Ac - Pc$) to 20 (Ap) years old matches those of the query bridge during the period from 35 ($Aq - Pq$) to 40 (now) years old. Therefore, the condition of the case bridge at age 25 years old (condition

rating = 3.25) can be used as a prediction for the future condition of the query bridge after 5 years from now, assuming that the two cases match in their deterioration factors.

The main motivation behind introducing the CBR approach in solving the problem of modeling bridge deterioration is to maximize the benefit of having a large amount of bridge data stored in BMS databases. Such databases can be seen as case libraries that contain thousands of cases that are continuously updated. Also, new cases are frequently added to these databases, which solves the problem of case accumulation that is the bottleneck of any CBR system. In addition, using CBR in modeling bridge deterioration has the following advantages:

1. The ability to perform "what-if" analysis for different maintenance scenarios by changing maintenance decisions and retrieving cases with matched decisions.
2. The use of specific knowledge encapsulated in previously experienced cases to predict the future condition of different bridge components with different construction materials.
3. The flexibility in accounting for unlimited amount of versatile deterioration factors, according to their availability, with no need for data segmentation.
4. The simplicity of model update that occurs as a natural by-product of case and data accumulation in the BMS database.
5. The capability of comparing the performance and maintenance actions of bridge components over an unbounded period of time depending on the availability of their historical data. This comparison minimizes the effect of the uncertainty and randomness of the deterioration process in condition predictions.

6. The ability to recognize the interaction among different bridge components by considering the condition of these interacting components as a part of the case description.
7. The flexibility in manipulating cases with different inspection periods and aggregated (i.e. continuous) condition ratings.

3.3 SYSTEM DEVELOPMENT

The proposed CBR system is a software system that has to be integrated with different modules of a bigger software system called BMS (refer to section 2.2), since it plays the role of the deterioration model in the host system. A system development process is a method for organizing all the activities related to creation, maintenance, and delivery of systems (Larman 1998). This definition applies for the development of any software system, regardless of its purpose (e.g. commercial, industrial, governmental, etc.) and whether it is a stand–alone or an integrated system. Phases followed generally in developing software systems will be applied to developing the proposed CBR system. At the macro-level, these phases as shown in Figure 3.5 are as follows (Larman 1998):

1. **Plan and Elaborate phase:** This phase is the initial conception of the software development in which different alternatives are investigated. It consists mainly of defining system requirements, system functionality, and the resources required to develop the system.
2. **Build phase:** This phase is an iterative phase that consists of multiple development cycles of analysis, design, implementation, and testing. These sub-phases are usually overlapped and dependent on each other. The main objective of the build

phase is to provide a working software system that correctly meets the system requirements and achieves the expected functionality defined in the previous phase.

3. **Deploy phase:** This phase is the last phase that brings the developed system into use (system delivery). This phase belongs to the development of commercial systems and, thus, it will not be covered in this research, since the developed system is a prototype system.

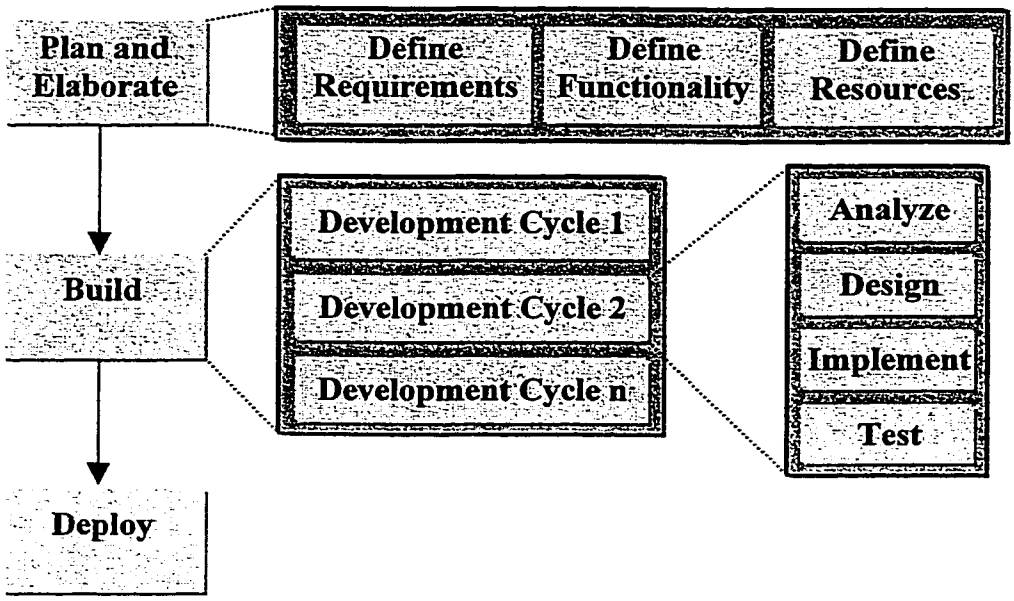


Figure 3.5: Phases of software system development (Larman 1998)

This section continues with a detailed discussion of the three components of the plan and elaborate phase: system requirements, functionality, and resources, while other development phases/sub-phases are presented in the following chapters. It should be noted that only the final development cycle, which is a result of several development iterations, is presented.

3.3.1 SYSTEM REQUIREMENTS

System requirements are necessary for the development of any software system to define its expected functionality. A list of system requirements was formulated to guide the development of the proposed CBR. This list describes the needs for that system and declares the necessary characteristics that should be manifested in its development. These requirements are:

1. **Hierarchical Decomposition of Bridges**: Bridges are complex structures that are composed of many components (e.g. sections, systems, assemblies, elements, etc.). Although it is rare to have two bridges that are completely identical, it is common to have components that are similar in different bridges. Therefore, the CBR system must be able to decompose bridges hierarchically into components that can be retrieved and reused individually (Rivard et al. 1998).
2. **Representation of Time-Dependent Data:** Bridge data can be categorized into static data (commonly referred to as inventory data), such as span length, material, structural system, etc., and dynamic data (commonly referred to as time-dependent data), such as material condition rating, performance condition rating, age, maintenance activities, etc. Time-dependent data are those data related to frequently changing features of a bridge (Kriviak 1999). The CBR system should be able to support the representation of time-dependent data as well as static data.
3. **Representation of Bridge Component Interaction:** The hierarchical decomposition of bridge components results in no relation among child components except through their common parent. This decomposition cannot represent the interaction among bridge components, such as girders, slabs, and bearings. Thus, the CBR system must

provide a way to represent the interaction among different bridge components in order to account for the effect of different deterioration mechanisms (Sianipar and Adams 1997).

4. **Data Reusing and Sharing:** In the bridge domain, many bridge components use the same attributes to describe their features. Moreover, some of these components use the same attribute values of other components. Thus, the proposed CBR system should be able to support reusing attribute and sharing attribute values among different bridge components, which saves storage space and facilitates efficient case representation (Howard et al. 1992).
5. **Versatile Similarity Measures:** The proposed CBR system has to measure the similarity between cases as the aggregation of the similarity between their static data, time-dependent data, and hierarchical decompositions. These similarities are greatly dependent on attribute types (numeric, symbolic, etc.). Moreover, time-dependent attributes require special consideration in measuring the similarity among their values. Hence, the CBR system should offer versatile techniques for measuring the similarity of different attribute types.
6. **Fuzziness of Retrieval Knowledge:** Retrieval knowledge consists mainly of attribute weights and their degrees of similarities. This knowledge is usually obtained from the domain experts who assess the relative importance of attributes and the relative similarity among attribute values based on their own experience and judgement. Therefore, the CBR system should support the representation of attribute weights and degrees of similarity as fuzzy number to account for the subjectivity of human experts (Sobanjo 1991).

7. **Case Adaptation:** The retrieval process rarely results in cases that exactly match the query case. Reusing solutions of partially matched cases without recognizing their unmatched features and how these features affect these solutions may reduce the system reliability (Roddis and Bocox 1997). So, adaptation capabilities are required to modify the proposed solution and eliminate the discrepancies between the query and retrieved cases.
8. **Accumulation of Bridge Data:** Only partial data about bridges, such as inventory data, are available initially, while other data, such as inspection and maintenance data, are accumulated as the bridge becomes older. Thus, the CBR system should support the addition, the removal, and the change of case contents in order to keep the case library always consistent and updated (Maher et al. 1995).
9. **Data Derivation:** The proposed CBR system should allow the derivation of new data items from stored ones. This capability eliminates data duplication (redundancy) that may often lead to inconsistency problems (Hannus et al. 1995). For instance, the age of a bridge beam at inspection should not be stored because it can be derived from its date of construction and its date of inspection.
10. **Modularity:** The CBR system should be structured in system modules each of which should also be broken down into separate sub-modules according to their functionality. This modularity guarantees manageable modifications to specific modules (Rivard et al. 1998; Hannus et al. 1995). For example, maintaining the system case library (e.g. addition, removal, and update of cases) should be provided in a module different from the modules of maintaining retrieval and adaptation

knowledge. This facilitates a quick and safe case accumulation without affecting the system knowledge.

11. **Compatibility:** BMS's have different modules and data repositories that contain the bridge data used in constructing the case Library. In order to introduce the proposed CBR system as a BMS module, this system has to be compatible with other BMS modules and their different data repositories. This compatibility facilitates the integration between different modules and allows a quick and accurate data transfer from and to the proposed CBR system.
12. **Versatility:** The CBR system should be able to handle different types of bridge data that are available in the state-of-the-art BMS's. Also, it should be able to represent bridges with various hierarchical decompositions, construction material, structural systems, geometrical configurations, and inspection and maintenance practices. Moreover, the CBR system has to be able to deal with different condition rating systems and variable inspection periods.
13. **Extensibility:** It is impossible to develop a system that satisfies every need for every user, but it is possible and recommended to develop a system that satisfies the current basic needs and allows future extensions to take place (Rivard et al. 1998). Therefore, the design of the proposed CBR system has to be flexible enough to allow the extensibility of the system in two dimensions:
    1. *Case structure dimension,* which means that the case hierarchical decomposition can be extended to represent new types of bridges or bridge components
    2. *Case content dimension,* which means that new case attributes can be added to describe new features such as new technologies in bridge inspection.

3.3.2 SYSTEM FUNCTIONALITY

System functionality, sometimes called system behavior, is documented in a Use Case Model. This model illustrates the system's intended functions, surroundings, boundaries, and internal and external relationships (Quatrani 1998). A description of the high-level Use Case Model developed for the proposed CBR system is presented here and shown in Figure 3.6.

1. **Actors:** They are not part of the system. They represent anyone (person) or anything (external system) that interacts with the system (Quatrani 1998). In Figure 3.6, an actor is represented as a stickman. Actors in the CBR system are: **BMS module or decision maker,** who consults the system to obtain predictions about the bridge condition for life-cycle cost calculations; **bridge engineer,** who maintains bridge inventory, inspection, and maintenance data; and **bridge expert,** who maintains the system retrieval and adaptation knowledge.
2. **Use Cases:** They model dialogues between the system and its actors and represent functionality provided by the system (Larman 1998). In Figure 3.6, a use case is represented as an oval. In the proposed CBR system, the defined use cases are as follows:
    i. Maintain Case Library: The purpose of this use case is to keep the system case library always consistent and updated. A bridge engineer activates this use case when storing new cases, deleting cases, or updating existing cases.
    ii. Maintain Retrieval Knowledge: The purpose of this use case is to improve the system performance by adjusting retrieval parameters. A bridge expert

activates this use case when defining attributes and their weights, similarity measures, and possible values.

iii. Maintain Adaptation Knowledge: The purpose of this use case is to improve the system adaptation capabilities by updating adaptation rules. A bridge expert activates this use case when storing new rules, deleting useless rules, or updating existing ones.

iv. Retrieve Cases: The purpose of this use case is to find cases from the case library that match the case at hand. This use case is activated when a BMS module or user inputs the features of the current problem and defines retrieval options. The system searches for similar cases and ranks them according to their matching score.

v. Adapt Case: The purpose of this use case is to modify the retrieved case to fit the current problem. This use case is activated when a BMS module or user needs to adapt one of the retrieved cases. The system assesses the differences between the selected case and the current one and applies appropriate rules to account for these differences.

vi. Validate User: The purpose of this use case is to verify that the password provided by the user is valid and it corresponds to the selected actor. This use case begins when the user activates any of the use cases presented above.

3. **Use Case Boundary:** This boundary identifies the system's responsibilities by determining what is internal versus external to the system (Larmen 1998). Only actors represent the external environment. The use case boundary is shown as a rectangle.

4. **Use Case Relationships:** There is only one type of relationships that may exist between an actor and a use case: association relationship. This relationship represents the communication between the actor and the use case, and its navigation direction represents who is initiating this communication. There are two types of relationships that may exist between use cases (Quatrani 1998): "Uses" and "Extends". "Uses" relationships are created between a Use Case and any other Use Case that uses its functionality. For example the "Retrieve Cases" and "Maintain Retrieval Knowledge" use the functionality of the "Validate User". "Extends" relationships are used to show an optional behavior that may run, based on actor selection. For example, the "Adapt Case" can be selected as an extension to the "Retrieve Cases".

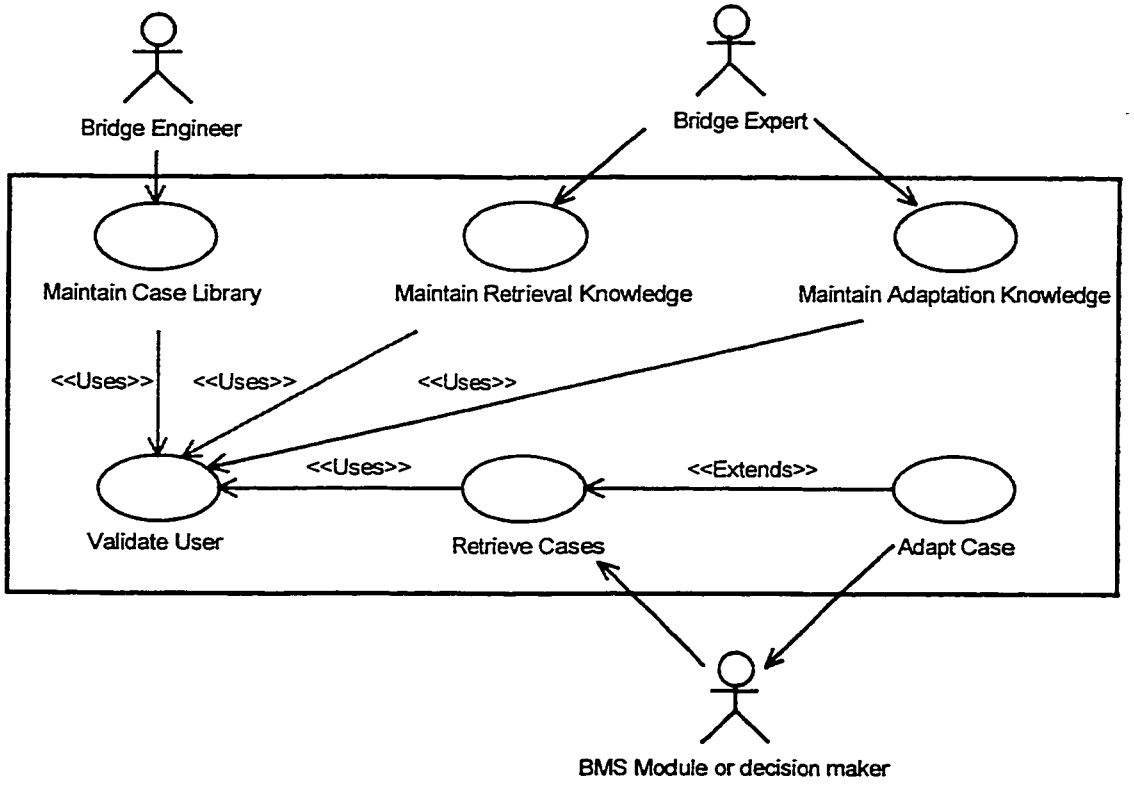


Figure 3.6: Use Case diagram of the proposed CBR system

3.3.3 SYSTEM RESOURCES

Software system resources are the tools through which the system design is mapped into real software components. The choice of these resources is a crucial step in the plan and elaborate phase because the following phases are greatly dependent on this choice (Larmen 1998). System requirements play an important role in the selection of the appropriate resources. In the proposed CBR system, two main resources have to be selected: the CBR shell used for generating a CBR application for modeling bridge deterioration and the Database Management System (DBMS) used for manipulating bridge records that constitute the case library. The choice of each of these resources is described in the following subsections.

3.3.3.1 CBR Shell

CBR shells are domain-independent tools with graphical user interface (GUI) to be used by non-programmer users in generating domain-dependent CBR applications (Althoff et al. 1995). Although CBR shells are domain-independent development tools, most of them are dedicated to solving a specific problem or performing a particular task that has special requirements and, consequently, to developing a distinct category of applications, such as design applications, planning applications, diagnosing applications, help-deck application etc. (Jaczynski and Trousse 98). In order to select the CBR shell that meets the system requirements presented earlier in section 3.3.1, a detailed investigation of the state-of-the-art CBR shells was carried out. This investigation started with reviewing the CBR shells used before in developing applications for the bridge domain and continued with exploring the commercially available CBR shells.

Reviewing the literature on infrastructure domains in general and the bridge domain in particular has resulted in several CBR shells that have been employed in developing a variety of CBR applications. Examples are: CASETOOL (Kumar and Krishnamoorthy 1995) and ART-1M (Edlund et al. 1994) used in developing CBR applications for bridge design; ART*Enterprise used in developing CBRidge Planner for planning of highway bridge projects (Tah et al. 1999); MEM-1 used in developing CB-BFX for resolving fabrication errors in steel highway bridges (Roddis and Bocox 1997); CasePlan used in developing CBR application for planning of power plant boilers (Dzeng and Tommelein 1995); Easy Haley Enterprise used in developing CACP for planning construction projects (Rankin et al. 1999); and the built up CBR shell used in developing CBR application for selecting retrofitting methods of fatigue damage on steel bridges (Tanaka et al. 1996). Most of these CBR shells are dedicated to design, planning, and diagnosing applications. The literature on CBR shells used for prediction tasks is limited to: ESTEEM, the shell used in developing a CBR application that predicts the outcome of construction litigation (Arditi and Tokdemir 1999); and REMIND, the shell that is used in developing a CBR application to predict the consequences of battle planning decisions (Goodman 1989).

Investigating the commercially available CBR shells has resulted in several products, some of them were not considered because they were developed mainly to satisfy the needs of business and management domains rather than engineering and manufacturing domains. CaseAdvisor (Sententia Software Inc.), CasePower (Inductive Solutions Inc.), RapidReasoner (Webpresence Technology), and KnowledgeBuilder (ServiceSoft

Corporation) are examples of such products. Only four CBR shells were candidates for further investigation. These products are KATE Tools (AcknoSoft Inc.), ART* Enterprise (Brightware Inc.), CBR Works 4 (TecInno GmbH), and Easy Reasoner (The Haley Enterprise Inc.). To evaluate these products, a list of criteria was established based on the system requirements listed in section 3.3.1 and the system developer's needs. Table 3.1 shows this list along with evaluation results.

Criteria	KATE Tools	ART Enterprise	CBR Works 4	Easy Reasoner
Object-oriented case representation	■	■	■	
Customized similarity assessment	■		■	■
Content and structure matching			■	
Adaptation with rules and functions		■	■	■
Integration with C++ applications	■	■		■
Database connectivity	■	■	■	■
Graphical User Interface	■	■	■	■
Windows 95/98 platform	■	■	■	■

Table 3.1: Evaluation of CBR candidate products

The most important criterion is the support for the object-oriented representation because this allows the representation of complex case structures and satisfies the "Hierarchical Decomposition of Bridges" requirement (requirement # 1 in section 3.3.1). The customized similarity assessment and the content and structure matching criteria allow the definition of new similarity measures and the comparison of both the hierarchical decomposition and contents of matched cases, which satisfies the "Versatile Similarity Measures" requirement (requirement # 5). The adaptation with rules and

functions criterion is necessary in order to adjust the retrieved cases for the unmatched parts with the query case and obtain more accurate solutions, which satisfies the "Case Adaptation" requirement (requirement # 7). Integration with C++ applications and connectivity with database systems are indispensable criteria in order to allow the communication with the different modules and data repositories of BMS's and satisfy the "Compatibility" requirement (requirements # 11). Having a graphical user interface and a Windows 95/98 platform criteria are required to, respectively, guarantee a friendly user interaction and avoid any incompatibility problems with the operating system. For more information about these products, refer to Appendix A.

The product evaluation shows that none of the candidate products satisfy all of the system requirements and developer's needs. CBR Works 4 was found to be the most suitable development tool especially if the intended system is a stand alone system and is not embedded in other applications. A more detailed investigation of CBR Works 4 development tool was carried out. This study disclosed some additional limitations pertaining to case representation and case retrieval aspects, which would hinder the development of the intended system. These limitations are (Tecinno 1999):

1. The object-oriented representation uses concepts that are equivalent to classes in object-oriented programming languages but without member functions.
2. In case of using a concept as a compound attribute (part-concept) of another concept (whole-concept), this part-concept cannot be a part of another whole-concept. This restriction affects the representation of time-dependent data (requirement # 2).

3. Attributes defined for any concept cannot be reused by another concept and their values cannot be shared by more than one object (requirement # 4).
4. Relationships between concepts are limited to one-to-many unidirectional relationships that cannot have different cardinalities (requirement # 3).
5. Customized similarity measures cannot handle complicated algorithms like the one needed to measure the similarity among time-dependent attributes (requirement # 5).
6. Attribute weights and degree of similarity are represented as crisp numbers rather than fuzzy numbers (requirement # 6). This representation is unrealistic for bridge management domains because these values are elicited from bridge experts based on their experience and judgement, which contain uncertainty.

Because of these limitations, a decision was made not to use a commercial CBR shell but to develop a new CBR system from scratch such that all system requirements and developer's needs would be satisfied. This system was implemented using the object-oriented programming language Microsoft Visual C++ 6.0 (VC++) along with the support of Microsoft Foundation Classes (MFC). Details about using these resources in the system implementation are presented in chapter 6.

3.3.3.2 Database Management System

In many CBR systems, cases are stored in a file that works as a case library. This file can be accessed by the reasoner (i.e. the software module used for both case retrieval and adaptation) for both read and write operations. During program execution, all cases stored

in that file are loaded into the primary memory to make the reasoner tightly coupled with cases. This architecture has some limitations. First, the size of the case library is restricted by the size of the available working memory. Second, the tight link between the case library and the reasoner prevents the sharing of the case library to other applications (e.g. BMS modules). Such architecture is not suitable for developing large CBR systems. Integrating CBR systems with database management systems (DBMS's) facilitates the sharing of the case library among multiple applications and allows the development of large CBR systems regardless of the size of the working memory (Ramarapu et al. 1997).

There are different types of DBMS's that can be integrated with CBR systems: Hierarchical Database Management Systems, Network Database Management Systems, Relational Database Management Systems (RDBMS's), and Object-Oriented Database Management Systems (OODBMS's). Each of these types is based on a different representational data model. The object data model has the potential to comprehend the features of hierarchical and network data models and provide a simpler data manipulation (Bertino and Martino 1993). Furthermore, object data model offers powerful features such as data nesting, encapsulation, inheritance, and polymorphism, which are not available in hierarchical and network data models.

RDBMS's have been the most popular DBMS's in recent years because of their capabilities in storing, indexing, searching, and querying large amounts of data. Technical aspects of both RDBMS's and OODBMS's were investigated in order to select the most appropriate type for developing the proposed system. This investigation

revealed some limitations of the relational data model that have been overcome in the object data model. These limitations are:

1. **Data Types and Structures:** Relational databases have a fixed set of data types and new data types cannot be defined by the user (POET 1998). However, object-oriented databases allow the combination of simple data types and encapsulating them into classes to formulate user-defined data types. Also, nested data structures facilitate the representation of complex real-life objects (Phan and Howard 1993).
2. **Modeling Relationships:** In relational databases, each record is represented as a row in a table, and the only way to represent its relationship with another record is to set its primary key in the other record as a foreign key. This way of modeling relationships is weak especially in representing many-to-many relationships and deep hierarchical decompositions (POET 1998). Object-oriented databases provide a natural and direct representation of relationships through the use of pointers that can represent relationships with different cardinalities (Wiegand and Adams 1995).
3. **Modeling Inheritance:** In relational databases, inheritance cannot be explicitly represented. Every level in a hierarchy requires a new table, and every program using the database must update every relevant table appropriately (POET 1998). In object-oriented database, class hierarchies are used to define powerful semantic concepts such as inheritance and polymorphism (Phan and Howard 1993).
4. **Grouping Code with Data:** In relational databases, there is no way to combine data with its related code. All data is globally accessible by any code (POET 1998). In object-oriented databases, both data and its related code are grouped into objects. This grouping defines specific routes for data access (Mullins 1994).

5. **Integration with Programming Languages:** Languages in relational database such as Structured Query Language (SQL) are limited to query operations only and do not have enough data manipulation capabilities. (POET 1998). Object-oriented databases and object-oriented programming languages are naturally integrated since they share the same model and data types (Mullins 1994).

Several commercial OODBMS's are currently available in the market. Different theoretical foundations, features, and consequently performances are expected in these products. Many studies have been carried out to compare these products. A comparison between eight of the most popular OODBMS's has been published in Amirbekian et al. (1997). This comparison is an updated version of a study called "Comparison of Ten OODBMS's" made by Kueng (1994). A summary of this update is included in Appendix A. Two of these eight OODBMS's listed in the latest study were eliminated because of the financial situation of their companies. To compare the suitability of the remaining six candidate products, a list of requirements was prepared. Table 3.2 shows these requirements along with the comparison results.

Criteria	OBJECTIVITY	OBJECTSTORE	POET	GEMSTONE	JASMINE	ITASCA
Pure object-oriented database.	■	■	■	■	■	■
Supporting C++ binding and applications programming	■	■	■		■	■
Supported by UML		■	■			
Supports the standard SQL		■	■		■	
Supports ad-hoc queries and updates with a GUI.	■	■	■	■	■	■
Access to other DBMS.	■	■	■	■	■	
Windows 95 platform.	■	■	■		■	■

Table 3.2: Comparison of OODBMS products (Amirbekian et al. 1997)

The most important criterion is being a pure object-oriented database. This means having support for all of the object-oriented features, such as encapsulation, inheritance, polymorphism, extensibility, etc. (Lafore 1991). Supporting C++ binding and application programming is essential for developing any software system (e.g. CBR system) that uses the OODBMS for data storage and data manipulation. Supporting the unified modeling language (UML) is also important for standardizing the notations used in designing the proposed CBR system. The support for standard SQL and ad-hoc queries criteria allows accessing and inquiring data within the database using various and efficient ways (Rivard 1994). The access to other DBMS's and having the Windows 95 platform criteria are necessary for satisfying the compatibility requirement and facilitating the integration with external systems and data repositories. The two OODBMS's ObjectStore and POET were found to be the best candidates with respect to these criteria and, consequently, for carrying out this research. ObjectStore was selected over POET because of its affordability.

CHAPTER 4

SYSTEM ANALYSIS

4.1 INTRODUCTION

Usually, a software system is overwhelmingly complex, and decomposing it into understandable chunks is essential to manage the complexity. System models are used to represent these chunks and to abstract essential aspects of the system (Rumbaugh 1997). There are two types of system models: analysis models and design models. Analysis models are those devoted to describing the domain and the problem space, while design models are those devoted to describing the solution and how it satisfies the system requirements (Larman 1998). One of the most important system analysis models is the conceptual model. A conceptual model, in general, is a representation of real-world concepts (physical or non-physical), in the problem domain, which are generally meant for domain personnel not for computer specialists (Elmasri and Navathe 1994). The conceptual model consists of domain concepts, their data, and the relationships among them (Rivard and Fenves 2000). Domain concepts result from decomposing the problem domain and using the terminology or the vocabulary of that domain to identify them.

This chapter describes the development of the conceptual model of bridges (i.e. the system analysis model). The chapter is organized as follows: first, the hierarchical decomposition of bridges that results in bridge concepts is presented; second, the data of the decomposed bridge concepts are described; and finally, the relationships among these concepts are established.

4.2 HIERARCHICAL DECOMPOSITION OF BRIDGES

Highway bridges are complex structures and representing them is not a simple task (Haque 1997). Decomposition, or divide and conquer, is a common strategy in dealing with complexity and identifying the concepts of complex domains (Larman 1998). In bridge management domains, bridge engineers and managers used to decompose bridges into components and store different bridge data at the level of these components. This way of data storage provides a good understanding for the features of each individual component along with its inspection and maintenance history. This understanding is necessary for the purpose of modeling bridge deterioration and, consequently, making efficient decisions regarding MR&R needs.

In Pontis, the most commonly used BMS in the U.S., bridge data are stored at three different levels (AASHTO 1999). Basic inventory data are stored at the structure level, which represents one highway structure, such as a bridge or a culvert. However, a structure may be divided into one or more structure units. Each structure unit represents a portion of that structure, such as a single span or a frame. Detailed inventory data about each portion are stored at the structure unit representing this portion. Each structure unit, in turn may be divided into one or more condition units. A condition unit represents any given element such as a pile, a column, a beam, or a bearing in a particular environment. Inspection data are stored at the level of condition units. Figure 4.1 shows an example of decomposing a truss bridge in Pontis. In this example, the bridge is decomposed into three spans (structure units), each span is decomposed into deck, bearings, columns, abutments, pier caps, bottom chord, etc. (condition units).

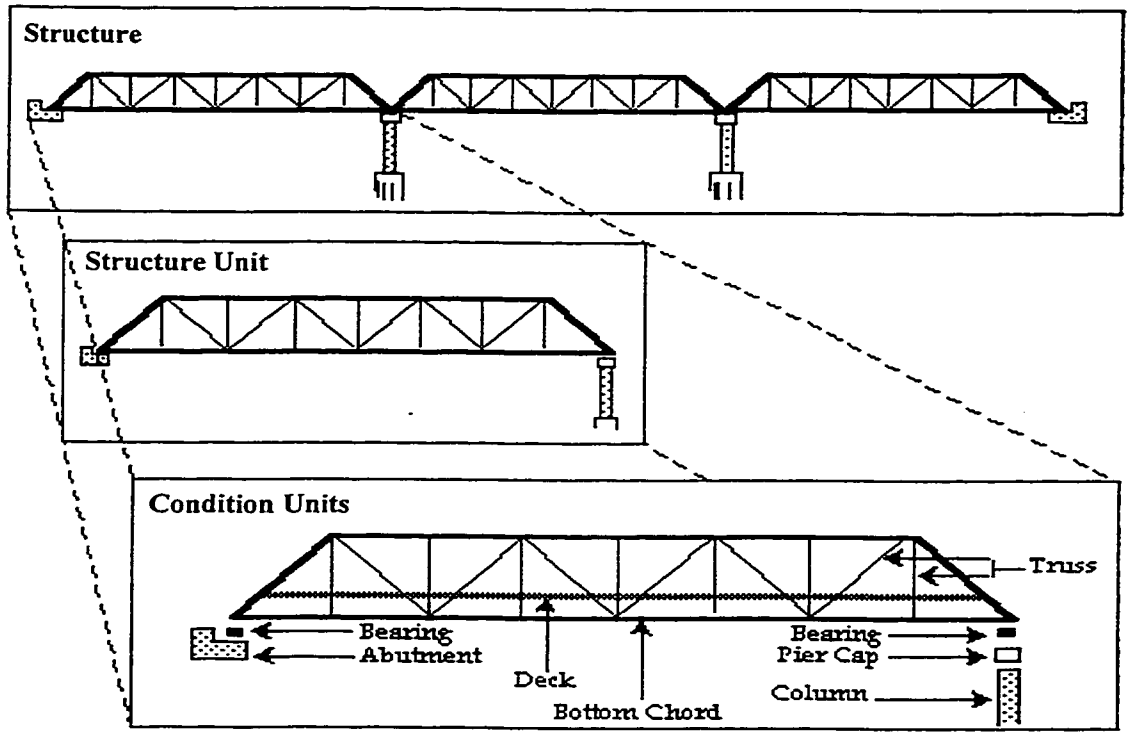


Figure 4.1: The decomposition of truss bridges used in Pontis (AASHTO 1999)

This way of decomposing bridges has three disadvantages. First, it separates the storage of inventory data (in structure units) from the storage of inspection data (in condition units). This separation is inconvenient because a combination of both types of data is needed in predicting the future condition of bridge components and in selecting the optimum MR&R actions (Kriviak 1999). Second, bridge components are decomposed into components (i.e. structure units) based on their location only and not based on their function, however, it is quite often aggregating information about the bridge components that perform the same functionality (MTQ 1997). For instance, in Pontis a bridge column and a bridge deck are decomposed from the same structure unit, because they are located in the same span, although they have different functionality and, consequently, belong to different structural systems: substructure and superstructure system respectively. Third,

system respectively. Third, decomposing a bridge into only three levels does not serve as an efficient abstraction mechanism that presents the different levels of granularity in dealing with bridge concepts (Rivard and Fenves 2000).

In order to identify the concepts required for building the conceptual model of bridges, bridges are decomposed in a different manner. This manner was inspired from the hierarchical decomposition of buildings proposed by Rivard and Fenves (2000). The preference of this hierarchical decomposition comes from its ability to eliminate the disadvantages presented above and its compatibility with the way bridge engineers decompose bridge for construction. This hierarchical decomposition follows the same logical order of breaking down bridge construction activities. It consists of seven levels of granularity that are shown in Figure 4.2 and described below.

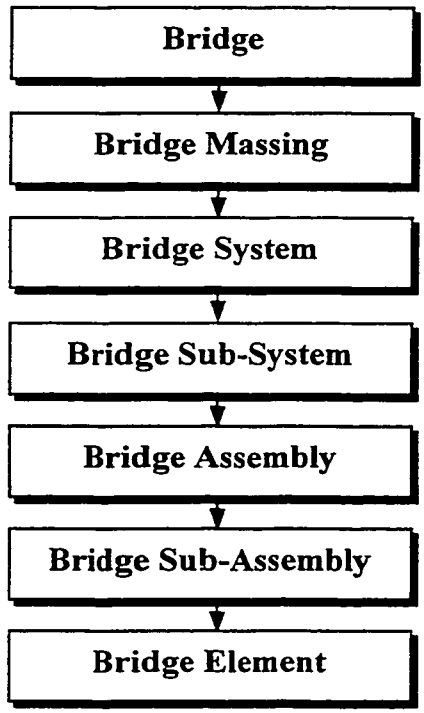


Figure 4.2: Levels of bridge decomposition

The root level is the structure level that represents a unique bridge structure. At the second level, a bridge structure is divided into bridge massings which represents parts of the bridge that are structurally different, such as the main bridge and bridge approaches, or constructed separately at different times, such as the main bridge, bridge ramps, and bridge extensions. A small bridge normally consists of three massings (i.e. the main bridge and two approaches), while a long bridge consists of many massings as shown in Figure 4.3. At the third level, each bridge massing is divided into only two bridge systems: structural system, which mainly maintains the bridge stability, and a non-structural system, which mainly maintains the bridge serviceability. At the fourth level, each bridge system is broken down into sub-systems according to their functionality. For example, the structural system is broken down into foundation, substructure, and superstructure sub-systems, while the non-structural system is broken down into joint, railing, drainage, protection, curb-sidewalk, and accessories sub-systems.

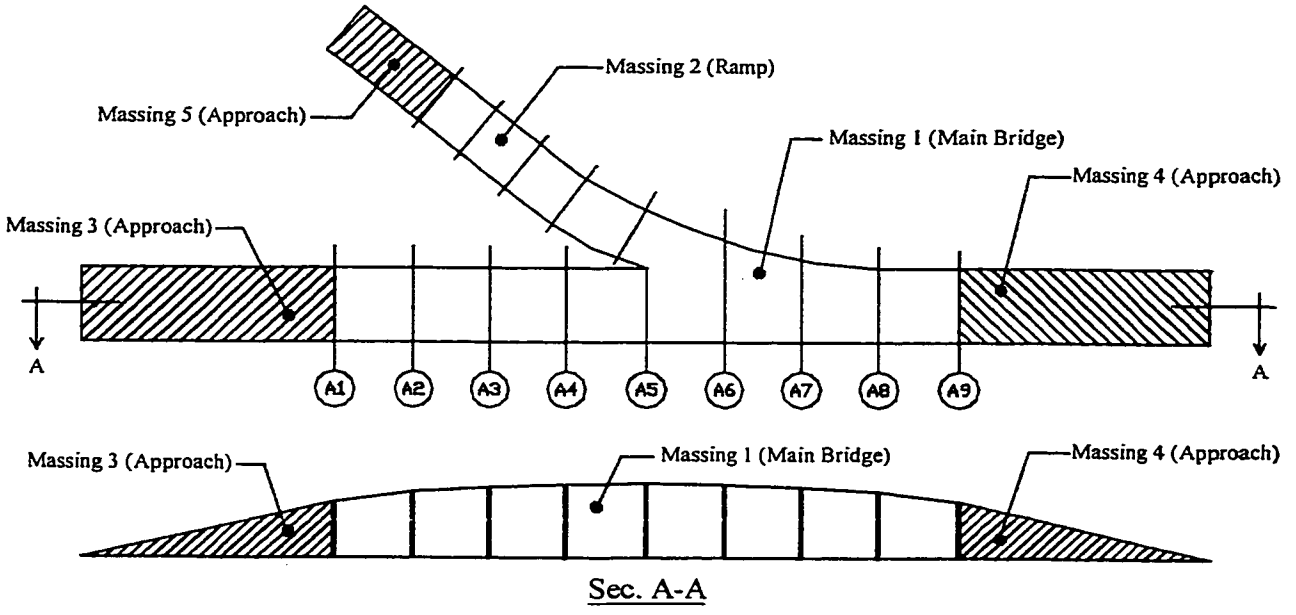


Figure 4.3: Examples of different bridge massings

At the fifth level, each sub-system is decomposed into bridge assemblies. A bridge assembly is a collection of bridge elements that are located together and physically connected. For instance, the bridge foundation sub-system may consist of many pile foundation and footing foundation assemblies, each pile foundation assembly consists of piles and a pile cap that are connected and located at one axis, while each footing foundation assembly consists of a footing and ground beams that are also connected and located at one axis. Figure 4.4 shows examples of bridge assemblies decomposed from bridge foundation, substructure, and superstructure sub-systems. Some complicated bridge assemblies need to be decomposed futher into sub-assemblies that constitute the sixth level of decomposition. A bridge sub-assembly is a smaller collection of bridge elements that participate in doing the same functionality and physically connected. For instance bracing elements, flooring elements, and main supporting elements are sub-assemblies of superstructure assemblies. Bridge elements are the lowest level in the hierarchical decomposition of bridges and represent the highest level of detail. A bridge element can be defined as the smallest bridge component that has its own inventory, inspection, and maintenance data. Examples of bridge elements are piles, columns, beams, truss members, joints, bearings, etc.

Figure 4.5 shows a part of the hierarchical decomposition of bridges that presents the seven levels of granularity. This part focuses on the hierarchy of the girder assembly in the bridge superstructure sub-system. A complete presentation of the developed hierarchical decomposition is presented in Appendix B.

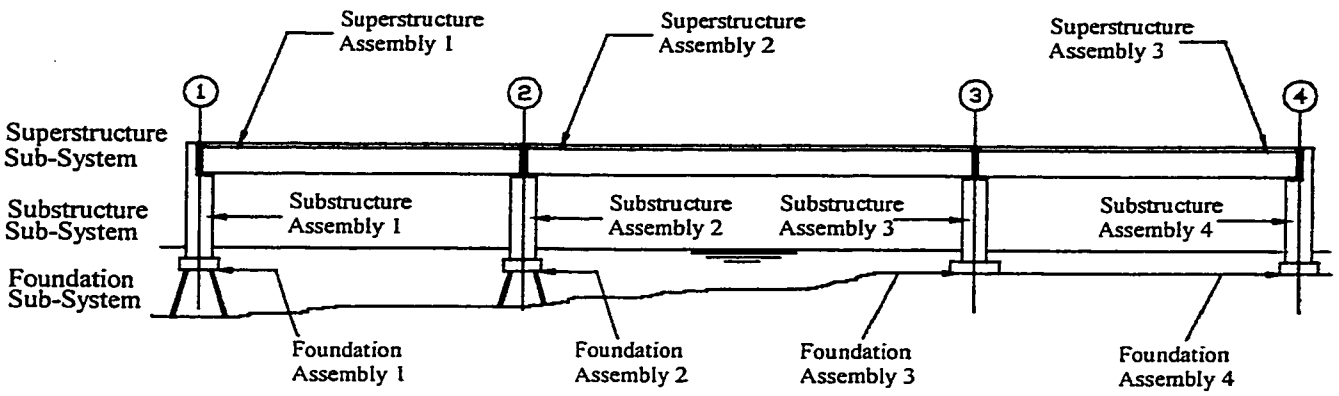


Figure 4.4: Examples of bridge sub-systems and assemblies

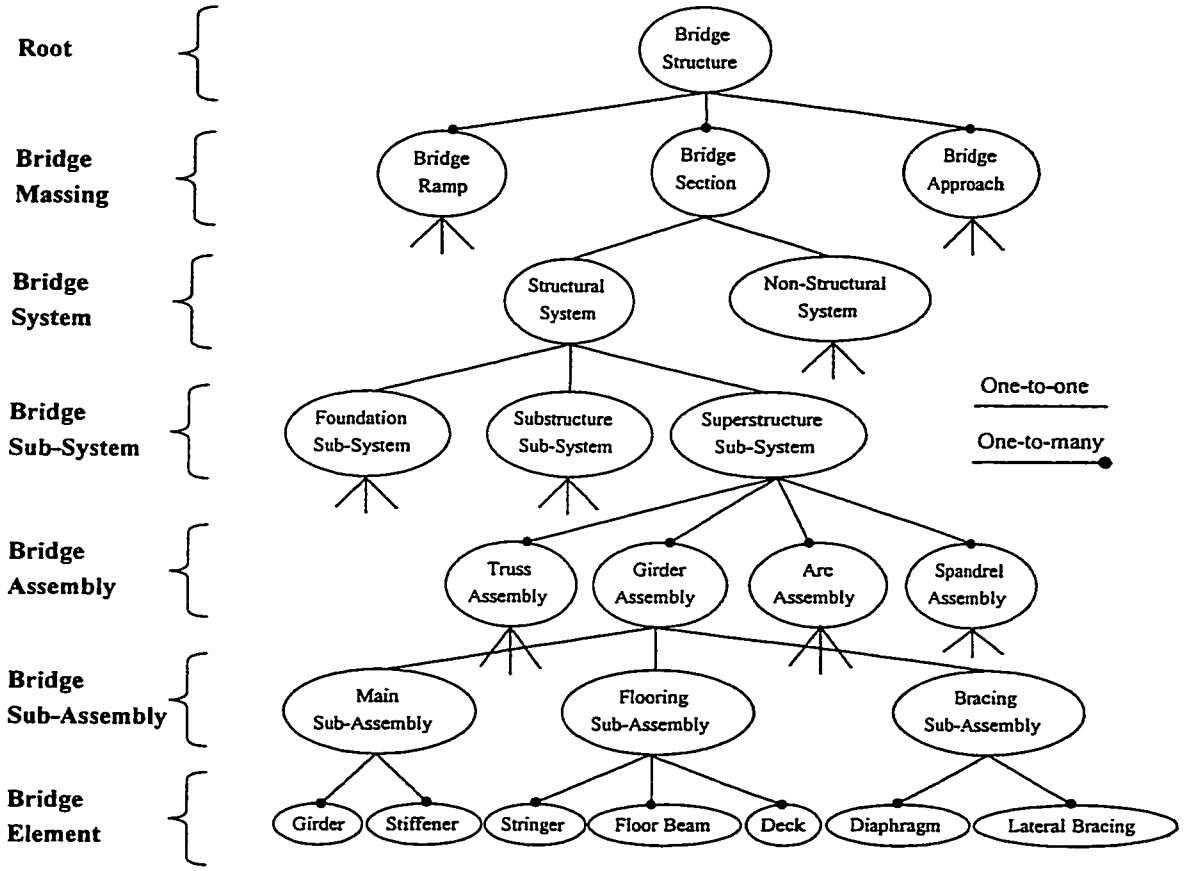


Figure 4.5: Example of the hierarchical decomposition of bridges

This hierarchical decomposition differs from the other decompositions used by the state-of-the-art BMS's in representing bridge structures. It breaks down bridge structures based on two criteria: function and location, which allows aggregating information about the bridge components that have the same location and those that have the same functionality (MTQ 1997). Also, this decomposition does not restrict the storage of bridge data to certain levels, but it supports the flexibility of storing different types of bridge data at any decomposition level (Kriviak 1999). Moreover, having seven levels of decomposition allows the abstraction of bridge concepts at different levels of granularity (Rivard and Fenves 2000).

4.2 BRIDGE DATA

Although the FHWA has listed all bridge data items required by a model BMS in its publication "Recording and Coding Guide for the Structure Inventory and Appraisal of the Nation's bridges" (FHWA 1995), none of the State DOT's has developed its BMS database identical to that guide. DOT's have devised their own procedures, codes, and databases. As evidence, a survey of bridge data needs and data collection practices in 14 States has shown great variation in both the number of data items collected per bridge, as shown in Figure 4.6, and the frequency of collecting these items, as shown in Figure 4.7 (Turner and Richardson 1994). The main reason for this variation is that the States have found that supplemental data are needed for their own unique environment especially for purposes such as deterioration modeling, maintenance decisions, and optimization of funds. In addition, individual States have sometimes been required by the FHWA to keep additional data.

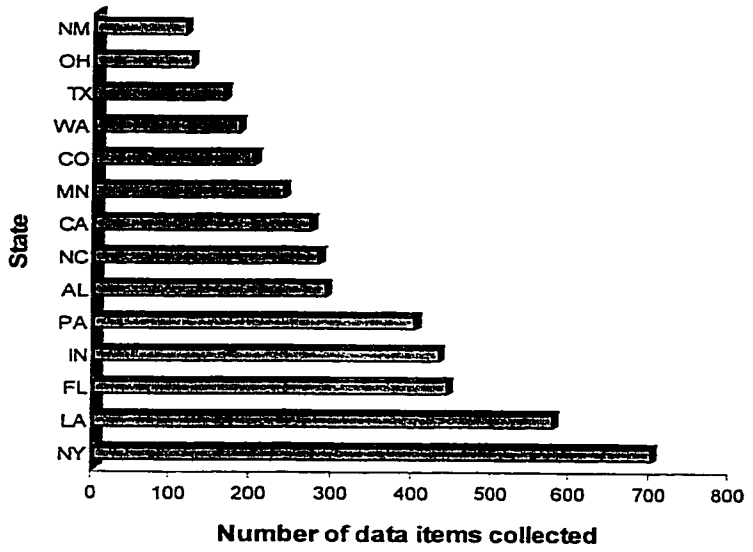


Figure 4.6: Number of data items collected in different states (Turner and Richardson 1994)

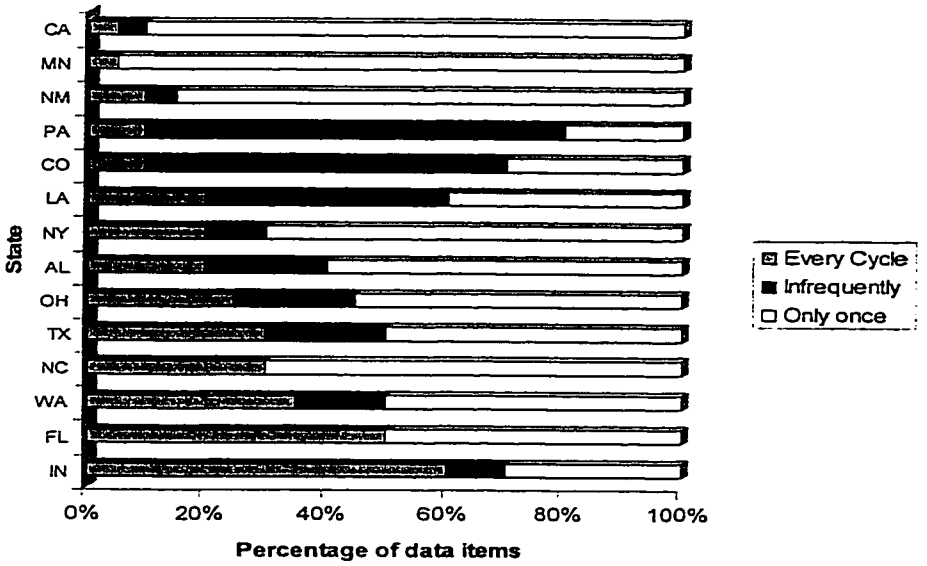


Figure 4.7: Frequency of data collection in different states (Turner and Richardson 1994)

In spite of this wide variation in the data items collected by the States DOT's, these data items can be grouped into categories and sub-categories as shown in Figure 4.8. These categories/sub-categories contain all types of bridge data that are required by a model BMS. For detailed information about these data items, refer to (Hudson et al. 1987; MTO 1991). The white boxes shown in Figure 4.8 indicate the data categories/sub-categories required for modeling bridge deterioration. A brief description of these data is provided below.

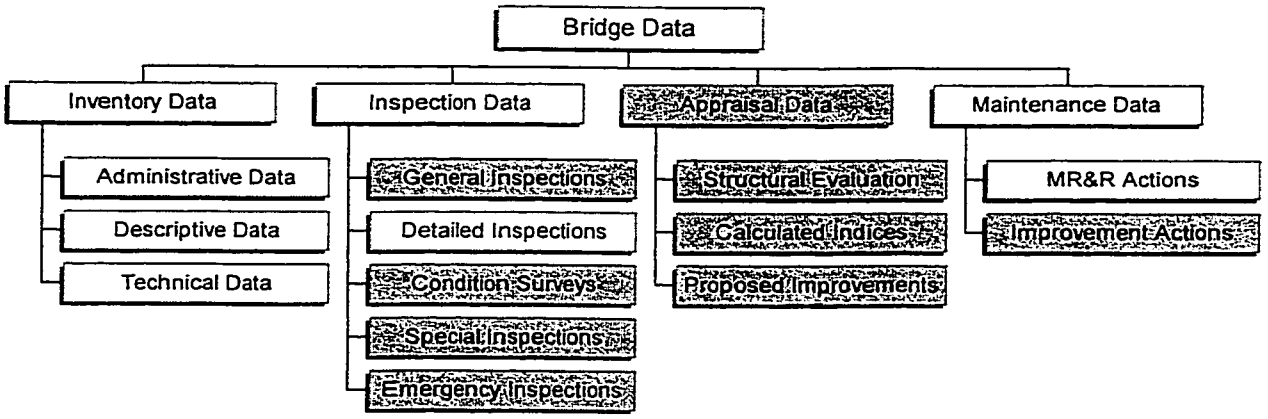


Figure 4.8: Categories and sub-categories of bridge data

Bridge inventory data are those data related to infrequently changing or unchanging bridge features (Kriviak 1999). Inventory data consist mainly of administrative, descriptive, and technical data. Administrative data include data items for bridge identification, location, essentiality, classification, and jurisdiction. Technical data include data items about the environment surrounding the bridge, traffic over and under the bridge, and postings on the bridge. Descriptive data include data items that describe the geometry, material, and structural system of different bridge components. They also include data about the various utilities on the bridge.

Detailed inspection data are collected frequently every inspection cycle. Inspection cycles differ from one transportation agency to another. In the U.S. and Ontario, bridges are inspected every two years, while in Quebec, bridges are inspected every three years. Bridge inspection data include administrative data about the inspection process itself, such as inspection date, inspection duration, inspection team, weather conditions, and equipment used; and condition ratings of all bridge elements. There are two types of condition ratings (MTO 1991): material condition rating (MCR) and performance condition rating (PCR). MCR of a component represents the condition of this component based upon the severity and extent of the observed defects in its material. PCR of a component describes the condition of this component based upon its ability to perform its intended function in the structure. The rating system used in evaluating the conditions of bridge components differs also from one transportation agency to another. In the U.S., a rating scale from 0 to 9, where 9 represents a new-bridge condition, is used (Madanat and Ibrahim 1995). In Ontario and Quebec, a rating scale from 1 to 6, where 6 represents a new-bridge condition, is used to evaluate both material and performance conditions as shown in Figure 4.9 and Table 4.1 respectively (MTO 1991).

		Guidelines for approximate reduction in the capacity of the component to perform its intended function		
Performance Condition Rating	Rating Description	Primary Component	Secondary Component	Auxiliary Component
9	Not Inspected			
6	Very good	0 to 1%	0 to 2%	0 to 5%
5	Good	1 to 5%	2 to 10%	5 to 20%
4	Fair	5 to 10%	10 to 20%	20 to 40%
3	Poor	10 to 15%	20 to 30%	40 to 60%
2	Urgent	15 to 20%	30 to 40%	60 to 80%
1	Critical	Over 20%	Over 40%	Over 80%
0	Non-Existing			

Table 4.1: Performance condition rating of components (MTO 1991)

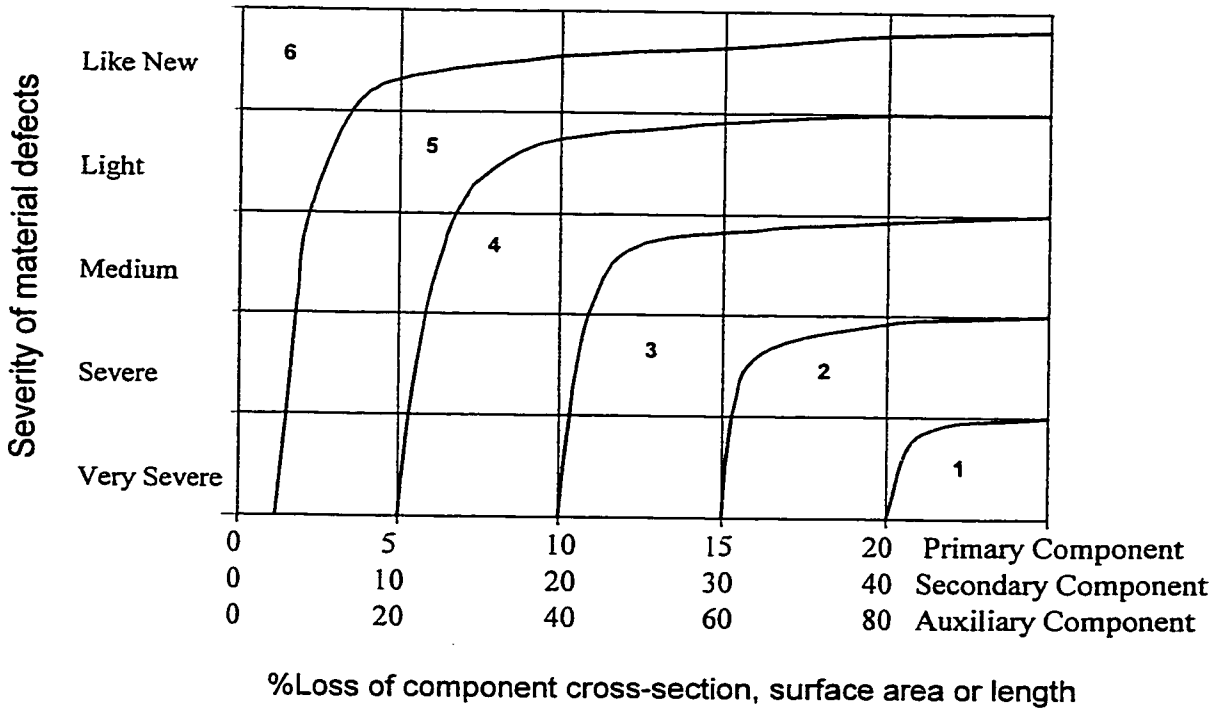


Figure 4.9: Material condition rating of components (MTO 1991)

Bridge MR&R data are collected only when bridges are maintained. These data include administrative data about the maintenance process itself, such as maintenance date, maintenance duration, contractor, weather conditions, and equipment used; and data about maintenance activities, such as the activity name, quantity, unit cost, and maintained components (Hudson et al. 1987).

The shaded boxes in Figure 4.8 indicate the data categories/sub-categories that are excluded from the proposed CBR system. The reason of excluding these data comes from their frequent unavailability, their irrelevance to the bridge deterioration problem, or both. For example, the structural evaluation of bridges is a good indicator for their deterioration but many bridges are rarely evaluated on a frequent basis. Calculated indices are data items derived from both bridge inventory and inspection data and, thus,

inspector regarding the recommended preservation and maintenance actions and they are not important for condition prediction as long as the actual preservation and maintenance actions are recorded. Improvement actions like deck widening and increasing vertical clearance represent the actions taken in order to improve the bridge level of serviceability and do not affect the bridge deterioration process. Bridge inspections other than the detailed visual inspection are either rarely available or devoted to particular bridge elements, such as underwater inspections.

4.3 CONCEPTUAL MODEL OF BRIDGES

Bridge concepts resulting from the hierarchical decomposition of bridges and their data presented in the previous section are not sufficient to define the conceptual model of bridges. Relationships among these concepts must also be identified. These relationships indicate some meaningful and realistic connections among bridge concepts. In order to represent these relationships and the concepts as well, a methodology or graphical notation such as the Unified Modeling Language (UML) is needed.

UML is "a language for specifying, visualizing, and constructing the artifacts of software systems" (Booch et al. 1997), where artifacts are diagrams and documents that comprise system models. UML is a comprehensive modeling language that can be used to represent system models during the whole system development process. UML standardizes artifacts and notations but does not standardize the development process (Larman 1998). A complete literature on the evolution of UML can be found in Fowler and Scott (1997). UML uses different types of diagrams at different development

phases/sub-phases, such as Use Case diagram, Static Structure diagram, Interaction diagram, Class diagram, State Transition diagram, Component diagram, Process diagram, and Deployment diagram.

Static Structure diagram is the diagram that represents concepts, data, and relationships of a conceptual model. This diagram is used in the system analysis sub-phase as a high level representation of the Class diagram used in the system design sub-phase discussed in the next chapter. The Process diagram will be used in Chapter 6 to represent the software components resulting from the system implementation. The description of the UML notations used in depicting a Static Structure diagram, a Class diagram, and a Process diagram is described below (Quatrani 1998; Fowler and Scott 1997). Examples of these notations are given in Figure 4.10.

- **Class:** A class represents a concept that has a structure and a behavior. A class is represented by compartmentalized rectangles. The top compartment contains the class name, the middle compartment contains class attributes, and the bottom compartment contains class member functions.
- **Stereotype:** A stereotype provides the capability to create new kind of modeling elements. For example, new kinds of classes such as entity classes, boundary classes, and control classes can be created. A stereotype is shown enclosed in (<<>>) above the class name.
- **Association Relationship:** An association relationship is a semantic connection between classes. It may be named by using a verb that represents the meaning of this connection. This verb reads correctly from left to right or top to bottom. An

association relationship may also be named by using the role name of the two connected classes. This type of relationship is depicted as a line connecting the associated classes.

- **Aggregation Relationship:** An aggregation relationship is a specialized form of association relationship in which a whole is related to its parts. Aggregation is known as a "part-of" or containment relationship. This containment may be by value or by reference. Containment by value implies exclusive ownership by the contained class. It is depicted as a line with a filled diamond next to the class denoting the whole. Containment by reference does not mandate exclusive ownership. It is depicted as a line with an open diamond next to the class denoting the whole.
- **Relationship Multiplicity:** Whether the relationship is association or aggregation, it should be attached with multiplicity indicators. Multiplicity defines the minimum and the maximum number of objects that participate in the relationship from the connected classes. Two multiplicity indicators should be written at the two ends of the relationship line.
- **Relationship Navigability:** Navigability of a relationship determines if it is a bi-directional or a unidirectional relationship. Unidirectional relationships are easier to implement and maintain while bi-directional relationships are more efficient. An open arrow is used at the end of the relationship line to show the direction of navigation in a unidirectional relationship.
- **Reflexive Relationship:** A reflexive relationship is used to communicate objects from the same class. This relationship may be association or aggregation.

- **Inheritance Relationship:** An inheritance relationship defines the relationship among classes when one class (subclass) shares the structure and/or behavior of another class (superclass). It is called "is-a" or "kind-of" relationship. Inheritance is represented by an arrow that starts at the subclass and points to the superclass. There are two ways to find inheritance: by generalization or by specialization.
- **Component:** A component represents a software file such as an executable or a dynamic-link library file. Components are related via dependency relationships that are represented by arrows starting at the dependent components.

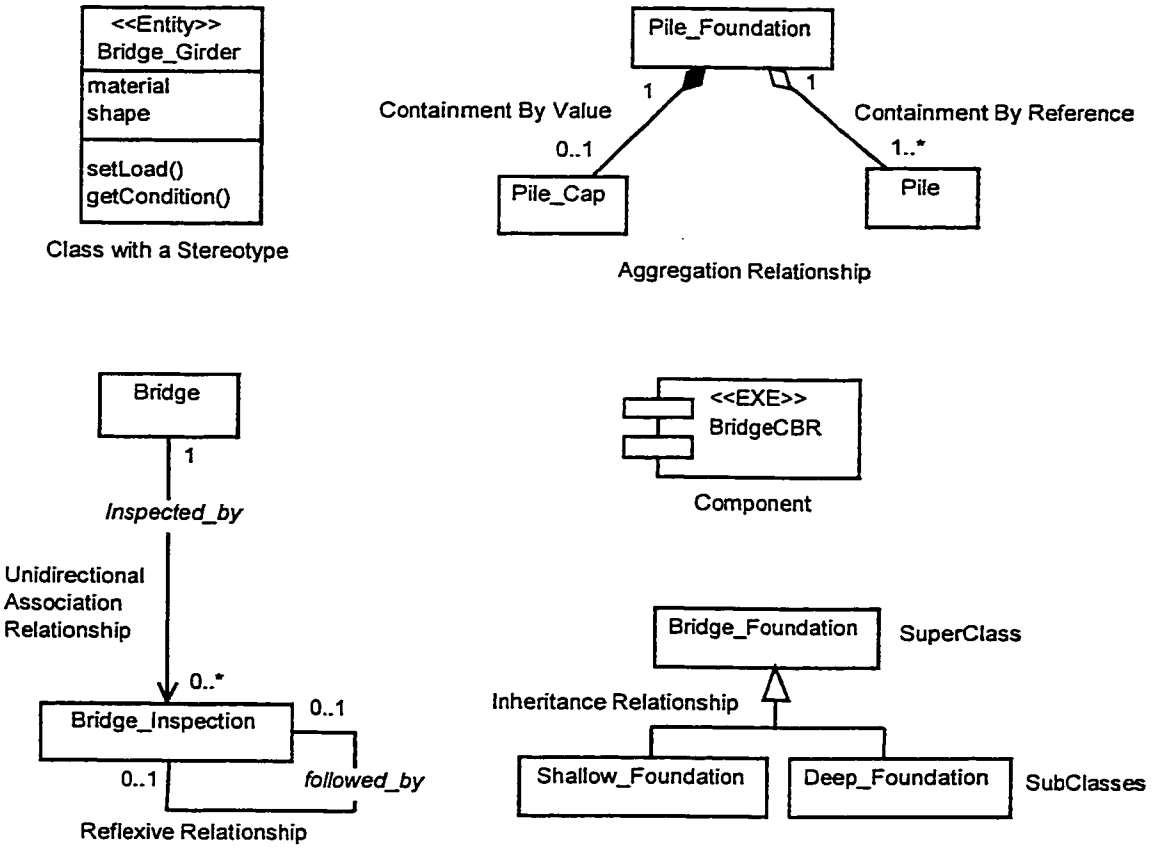


Figure 4.10: UML notations used in Static Structure, Class, and Process diagrams

Figure 4.11 shows the skeleton of the conceptual model of bridges that represents both the highest and lowest levels of the hierarchical decomposition of bridges along with the concepts representing bridge inspection and maintenance data. In this figure, a concept called "Bridge" is used to represent a complete bridge structure. This concept is the root for all data pertinent for a given bridge. The "Bridge" concept has association relationships with both "Bridge_Inspection" and "Bridge_Maintenance" concepts that represent, respectively, a bridge visual inspection activity and a bridge maintenance contract. A concept called "Bridge_Element" is introduced to be a superclass for all types of bridge elements. This concept has association relationships with both "Element_Condition" and "Maintenance_Action" concepts that represent, respectively, the condition ratings and the MR&R activity for a specific bridge element.

Figure 4.12 shows the part of the conceptual model that represents bridge concepts resulting from the hierarchical decomposition of bridges to the level of bridge assemblies. This part focuses on the structural system of bridge sections only. Figures 4.13, 4.14, and 4.15 show, respectively, parts of the conceptual model that represent bridge concepts resulting from the hierarchical decomposition of foundation assemblies, substructure assemblies, and superstructure assemblies to the level of bridge elements. All types of bridge elements are specialized from the "Bridge_Element" concept in order to have the same relationships with "Element_Condition" and "Maintenance_Action" concepts. Additional concepts can be specialized from these two concepts to represent different ways of inspecting and maintaining particular bridge elements, such as "Deck_Condition", "Joint_Condition", and "Bearing_Maintenance_Action".

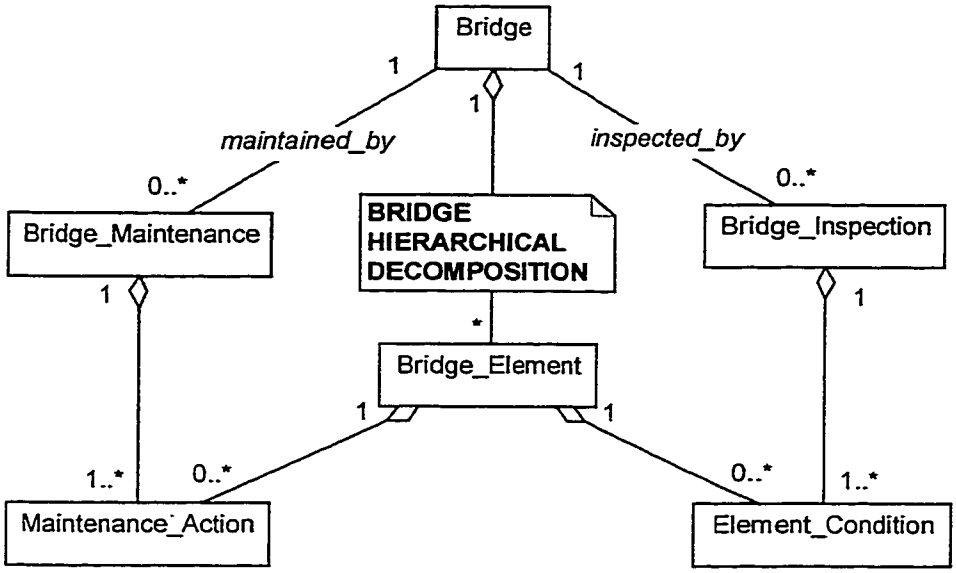


Figure 4.11: The skeleton of the conceptual model of bridges

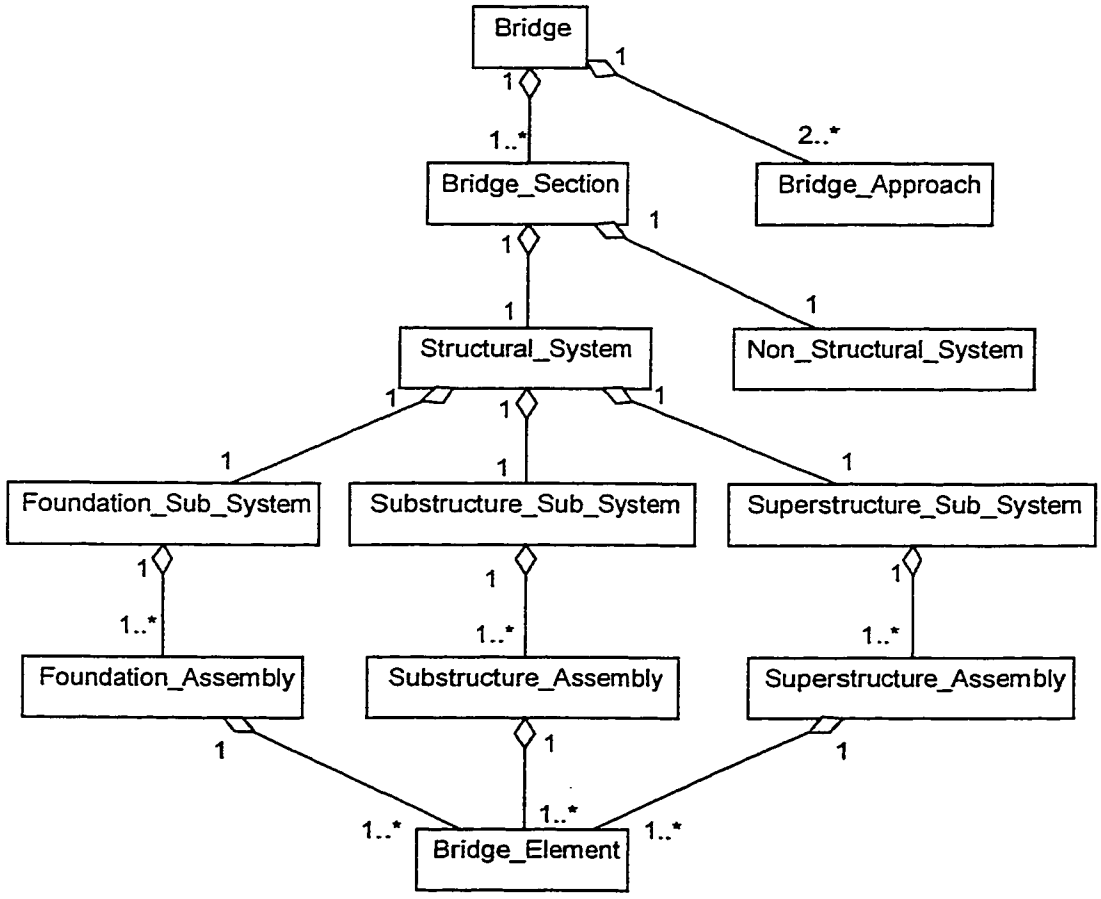


Figure 4.12: The conceptual model of bridges to the level of bridge assemblies

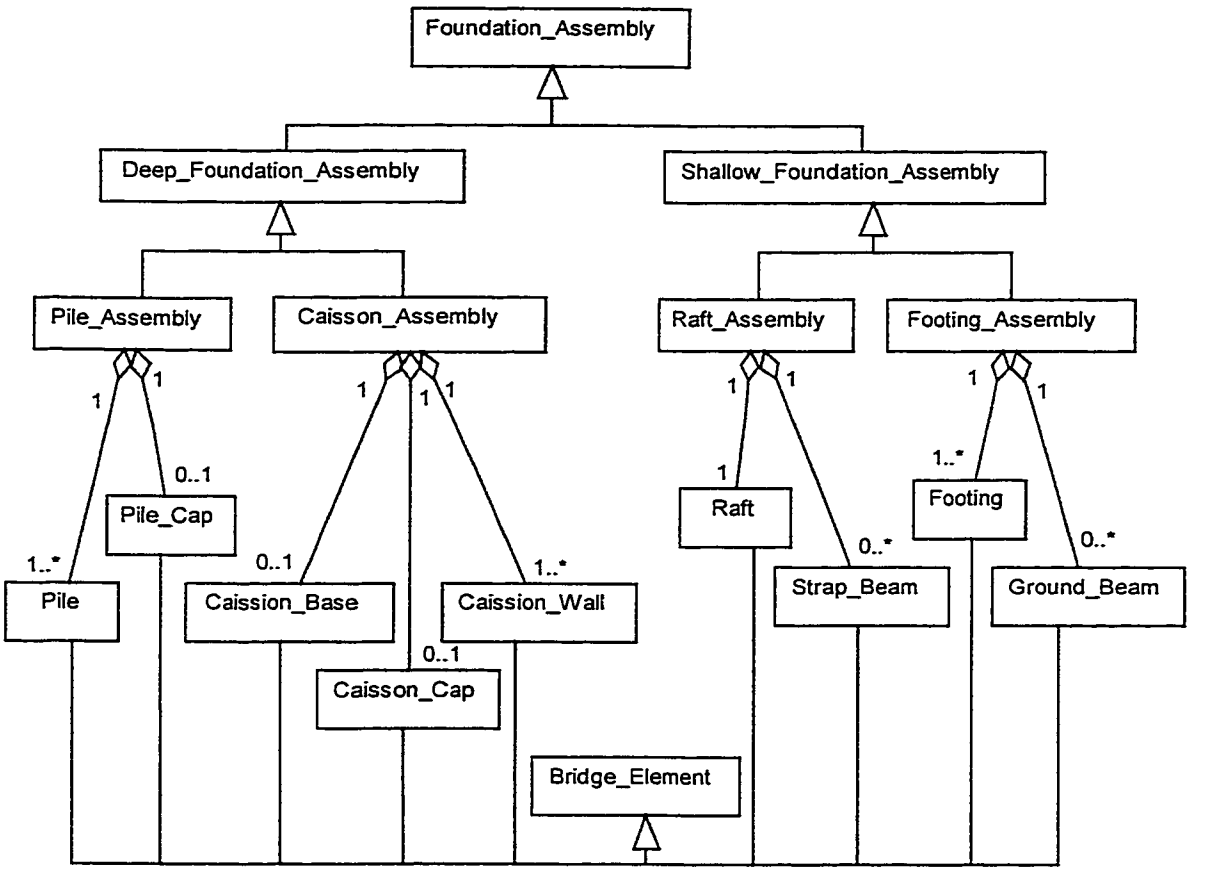


Figure 4.13: The conceptual model of the foundation assemblies

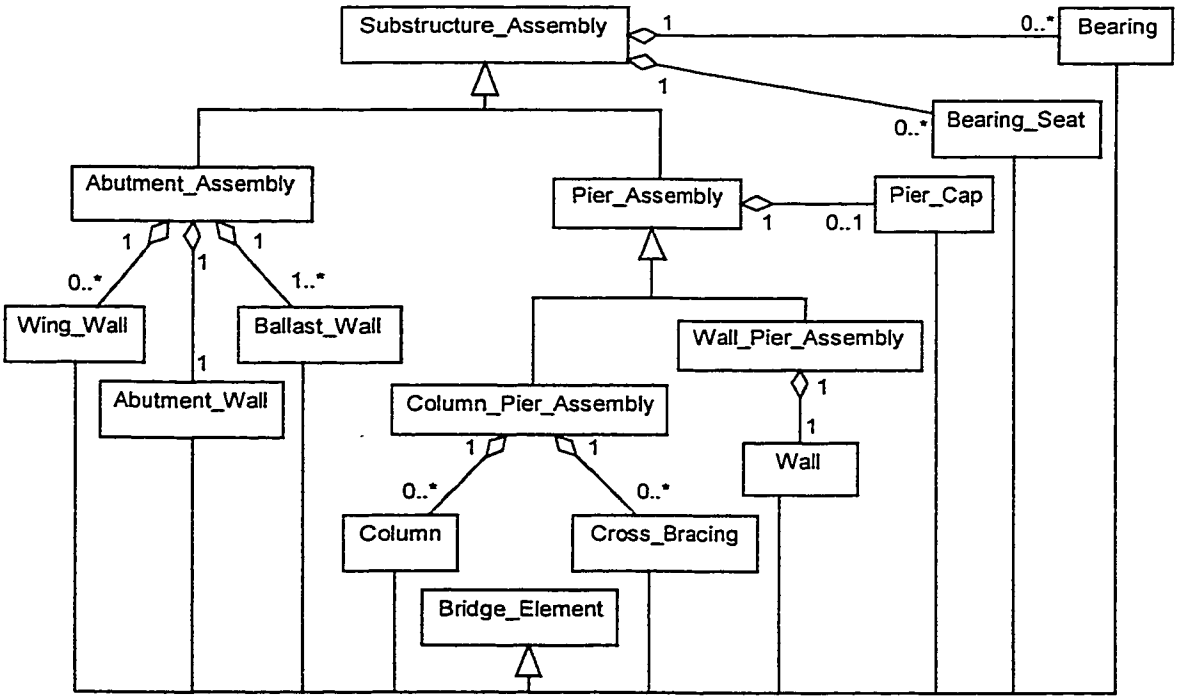


Figure 4.14: The conceptual model of the substructure assemblies

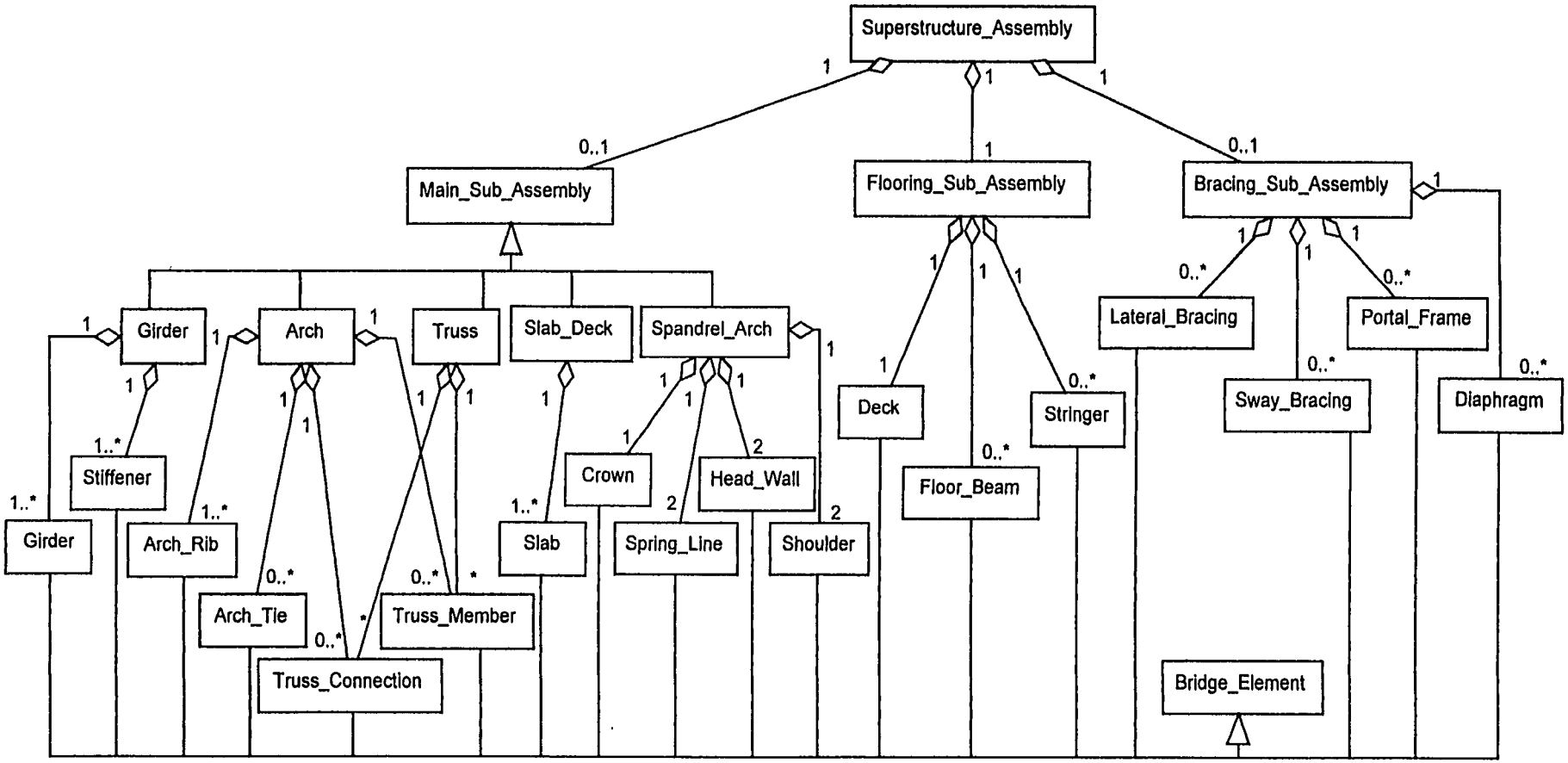


Figure 4.15: The conceptual model of the superstructure assemblies

CHAPTER 5

SYSTEM DESIGN

5.1 INTRODUCTION

The conceptual model resulting from the system analysis sub-phase of the build phase identified the domain concepts, their data (i.e. attributes), and relationships. In the system design sub-phase, the system development cycle moves towards describing the logical solution using a design model. This model consists of object-oriented classes, data members, and member functions (Larman 1998). The conventional methodology for going from analysis to design consists in transforming domain concepts into classes, attributes into data members, relationships into data members and member functions. This transformation is not a direct transformation, but some domain concepts may be merged, while others may be discarded. However, the final design model should have a similar structure to the conceptual model.

Transforming the conceptual model presented in Chapter 4 into an object-oriented design model is inappropriate for the proposed system due to three limitations. First, this conceptual model has a large number of bridge concepts even though it was limited to only the concepts of the structural system of bridge sections. Representing each of these concepts as an individual class in an object-oriented design model will lead to a very complicated design model (i.e. plenty of classes, data members, and member functions) that results in a heavy coding burden. Second, representing each bridge concept as a class does not satisfy the system versatility requirement as the system will be limited to only

the bridge concepts presented in its design model. These concepts differ from one BMS to another depending on the types of bridges covered by that BMS and the types of data collected about these bridges (refer to section 4.2). Third, this transformation does not satisfy the system extensibility requirement, because once the design model is implemented no more concepts can be added and the system becomes rigid for any future extensions.

In order to eliminate the limitations described above, a CBR shell that can be used for modeling the deterioration of any infrastructure facility (e.g. bridges, culverts, buildings, pavements, etc.) will be developed instead. This shell represents a CBR system that is generic and flexible enough to apply the CBR approach for modeling facility deterioration in a domain-independent manner. In order to use this shell, the user has to construct, at run-time, the so-called domain model. A domain model is the conceptual model of the domain under consideration. It consists of the domain concepts, their attributes, and the relationships among them (refer to section 4.3). This model differs from a agency to another and from an infrastructure management system (IMS) to another. The main function of such a model is to define how cases are represented and accumulated in the case library. Once the domain model is defined and the relevant cases are created, none of its concepts or its relationships can be removed, but more concepts can be added and new relationships can be established. Although a CBR shell requires more time and effort from the user in building a domain model, it provides the user with a highly reusable, extensible, and versatile CBR system.

In the rest of this chapter, the architecture of the CBR shell that satisfies the requirements of the proposed system is defined. This is followed by a description of the design model for each aspect of the intended CBR shell.

5.2 SYSTEM ARCHITECTURE

Designing a CBR shell is not a simple task because a CBR shell has to support the four aspects of CBR systems (case representation, accumulation, retrieval, and adaptation) in a domain-independent manner. Modularity is one of the essential requirements in designing a CBR shell. Modularity means structuring the shell into modules, each of which deals with a specific functionality, that resolve the complexity of the shell design and provide manageable system maintenance. Thus, the proposed CBR shell has been divided into three main modules as shown in white boxes in Figure 5.1.

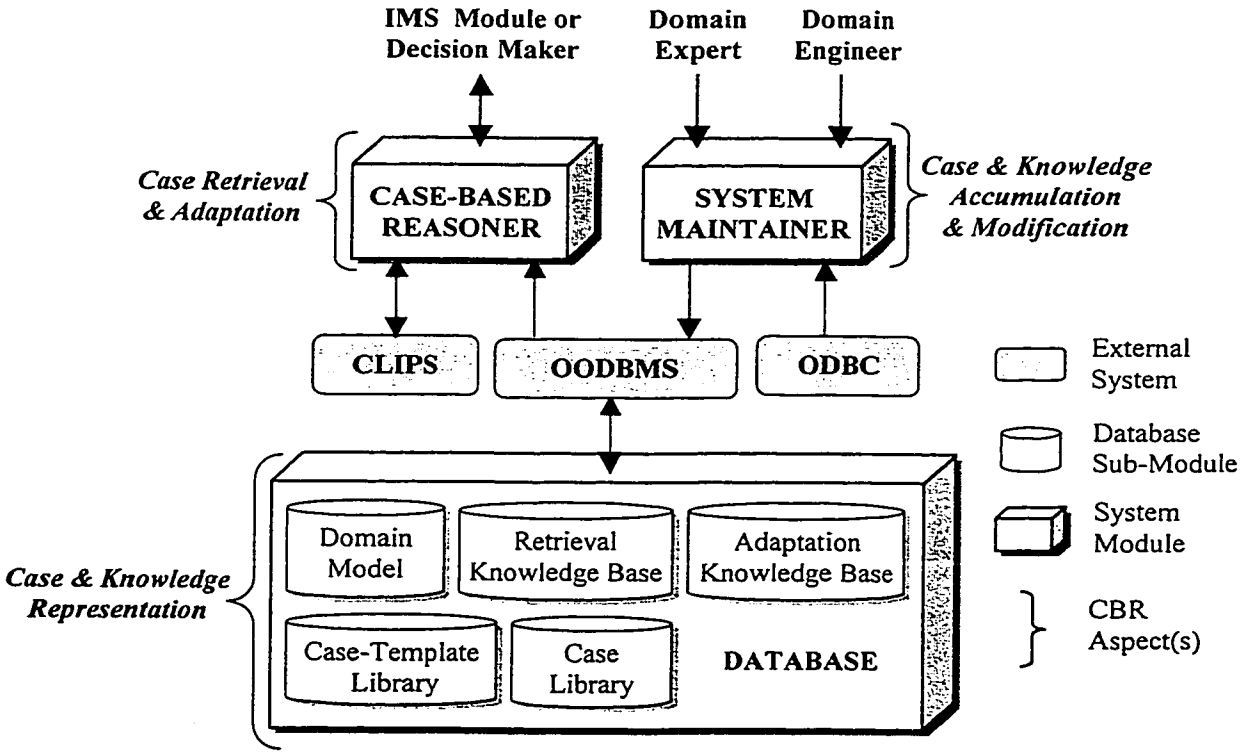


Figure 5.1: The architecture of the proposed CBR shell

The CBR shell modules are as follows:

a - **Case-Based Reasoner module:** This module is concerned with the case retrieval and case adaptation aspects. It consists in a retrieval algorithm to retrieve the cases that best match the query case. It also communicates with the inference engine of the expert system programming language CLIPS to adapt the best-matched case and provide an adapted solution.

b - **System Maintainer module:** This module is concerned with the case accumulation and modification aspect. It handles all maintenance operations, such as addition, removal, and update, in all database sub-modules. It also communicates with the Object Database Connectivity (ODBC) to import data from other data sources.

c - **Database module:** This module is concerned with the case and knowledge representation aspect. It works under the control of the OODBMS ObjectStore 6.0 that provides the persistent storage and data management capabilities. This module consists of five sub-modules, each of which is designed to store different types of system data. These sub-modules are as follows:

1. **Domain Model sub-module:** It stores the concepts and the relationships that constitute the domain model.
2. **Retrieval Knowledge Base sub-module:** It stores attributes that belong to each domain concept defined in the domain model. It also stores the knowledge about these attributes which is required for the retrieval process. This knowledge consists of attribute roles, weights, degrees of similarity, etc.
3. **Adaptation Knowledge Base sub-module:** It stores the domain knowledge elicited from domain experts (or the literature) in the form of IF-THEN rules.

These rules are transformed by the Case-Based Reasoner into CLIPS constructs that are manipulated later during the case adaptation process.

4. **Case Template Library sub-module:** It stores the structure of some cases that are common and can be used as templates to speed up the creation of new cases.
5. **Case Library sub-module:** It is the core sub-module of the CBR shell. It stores the structure and data of each case. This data will be used in the reasoning process to provide the user with the problem solution.

In Figure 5.1, system maintenance operations are available to both domain expert and domain engineer actors. A domain expert is the actor that has the authority to add, remove, and update knowledge in any of the first three database sub-modules (Domain Model, Retrieval Knowledge Base, and Adaptation Knowledge Base). A domain engineer is the actor that has the authority to add, remove, and update cases in the other database sub-modules (Case Template Library and Case Library). An IMS module or decision maker is the actor whose authority is limited to consulting the system through the Case-Based Reasoner module. This actor has a bi-directional communication with the Case-Based Reasoner module because it feeds the Case-Based Reasoner with an input case and receives from it the results of case retrieval and adaptation processes. The bridge expert, bridge engineer, and BMS module or decision maker actors of the Use Case diagram shown in Figure 3.6 are examples of a domain expert, domain engineer, and IMS module or decision maker respectively. It should be noted that the Use Case diagram that represents the functionality of the CBR shell is the same as the one shown in Figure 3.6,

except it has an additional use case called "Maintain Domain Model" that is activated by the domain expert actor to add, remove, or update concepts in a particular domain model.

The System Maintainer module and the Case-Based Reasoner module can access any of the database sub-modules through the OODBMS. Due to the fact that the main function of the System Maintainer module is to apply basic database maintenance operations such as add, delete, or update, its link with the OODBMS is represented by one arrow pointing downward. Because the Case-Based Reasoner module only accesses the data in the database sub-modules without modifying them, its link with the OODBMS is represented by one arrow pointing upward. The Case-Based Reasoner module also has a bi-directional link with the inference engine of CLIPS because it supplies CLIPS with retrieved cases and receives from it adapted cases, while the System Maintainer module has a unidirectional link with the ODBC because it only imports data from it.

In the following sections, the persistent part (i.e. the part used in storing persistent data) of the CBR shell design model will be presented in stages, each of which deals with one aspect of CBR systems. These stages are organized as follows: case representation, case retrieval, case adaptation, and case accumulation and modification.

5.3 CASE REPRESENTATION

Case representation is the essence of designing CBR systems because case representation organizes the information used in describing the problem and its solution (Kolodner 1993). Cases may be simple in their structure and not contain any sub-cases.

Such cases can be represented efficiently using simple representational paradigms, such as attribute value pairs or text descriptions. Cases may also be complex in their structure and composed of many sub-cases, which is the situation in infrastructure facility domains (refer to the hierarchical decomposition of bridges in section 4.2). Such complex cases have to be represented using a powerful representational paradigm such as the object-oriented paradigm that supports case/sub-case hierarchies (Maher et. al. 1995).

In conventional CBR systems, which are domain-dependent, the design model reflects clearly how cases are represented, while in CBR shells, which are domain-independent, the design model is very abstract and does not show clearly how cases are represented. In the latter situation, the function of the design model is to allow the representation of different domain models, each of which depicts clearly how cases from a specific domain are represented. Figure 5.2 shows the way cases are represented in CBR shells. The following sub-sections show the details of how domain models are represented and how they are used to represent cases in their domains with illustrative examples from the bridge management domain.

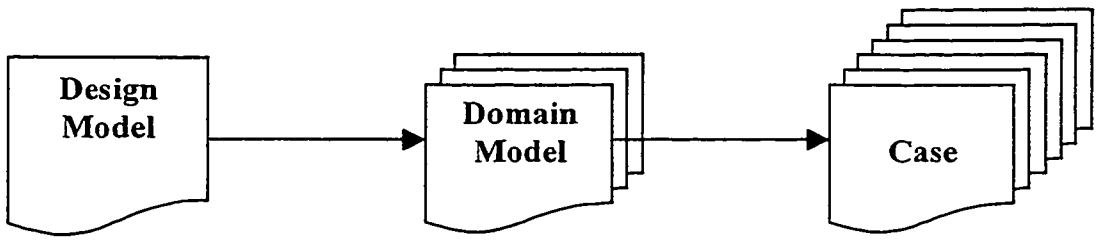


Figure 5.2: Case representation in CBR shells

5.3.1 DOMAIN ENTITIES

In the developed CBR shell, a case from a specific domain is represented as a collection of interrelated domain entities (Rivard and Fenves 2000). A domain entity is defined as a container that stores all the information about a component with a physical existence in an infrastructure facility. This component may be as complex as a bridge, building, or culvert; or as simple as a pile, beam, wall, or slab. Using the object-oriented representational paradigm, each domain entity is represented as an object instantiated from a class called "Domain_Entity". Each domain entity object has four data members as presented by the class diagram shown in Figure 5.3. The "id" data member is used to store a non-null and unique identifier for the domain entity such as a bridge number, column number, etc. The "name" data member is used to store a classification for the domain entity such as pile foundation assembly, footing foundation assembly, raft foundation assembly, etc. This way of classifying domain entities is used whenever a specialization of entities is needed (refer to the conceptual model shown in Figure 4.13). The "description" data member is used to store any textual remarks about the entity. The "status" data member is a Boolean field that determines if the entity is existing or demolished. This is because demolished entities still represent cases that can potentially be retrieved and reused.

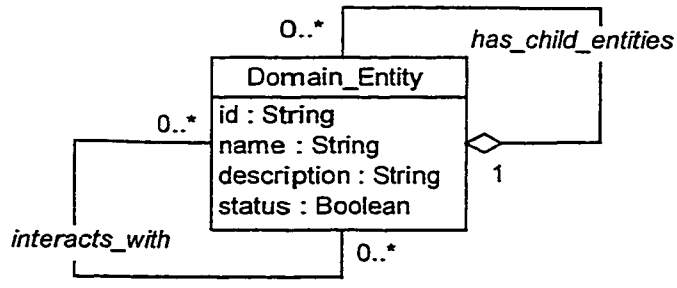


Figure 5.3: Representation of domain entities

In order to support the representation of complex case structures, the "Domain_Entity" class has a reflexive bi-directional one-to-many aggregation relationship called "has_child_entities". This relationship type allows a domain entity to be made of any number of other domain entities. For instance, a foundation assembly can be represented as a domain entity that contains other domain entities representing the pile cap and piles of that assembly. This aggregation relationship facilitates the hierarchical decomposition of cases into sub-cases with an unlimited number of decomposition levels.

Domain entities may have another type of relationship among them, which is different from the aggregation relationship. This type of relationship represents the interaction among the different components of the represented facility. Therefore, another reflexive bi-directional many-to-many association relationship called "interacts_with" has been added to the "Domain_Entity" class as shown in Figure 5.3. With this relationship, every domain entity can identify the other domain entities that it interacts with. For example a bridge slab domain entity identifies domain entities such as the deck joints, the end bearings, and the supporting girders as its interacting entities.

5.3.2 DOMAIN ENTITY TYPES

Many domain entities have common features among them and common types of relationships with other domain entities. Defining these features and types of relationships for every individual domain entity is inconvenient (Elmasri and Navathe 1994). It is more logical and convenient to define these features and types of relationships for groups of domain entities such as bridge columns, girders, foundation assemblies, and

so on. Therefore, a class called "Domain_Entity_Type" has been introduced as shown in Figure 5.4. Each object instantiated from that class represents a group of domain entities that have the same characteristics. Each domain entity type object has two data members: a "name" data member to store a non-null and unique name for that domain entity type and a "description" data member to store a narrative description of it.

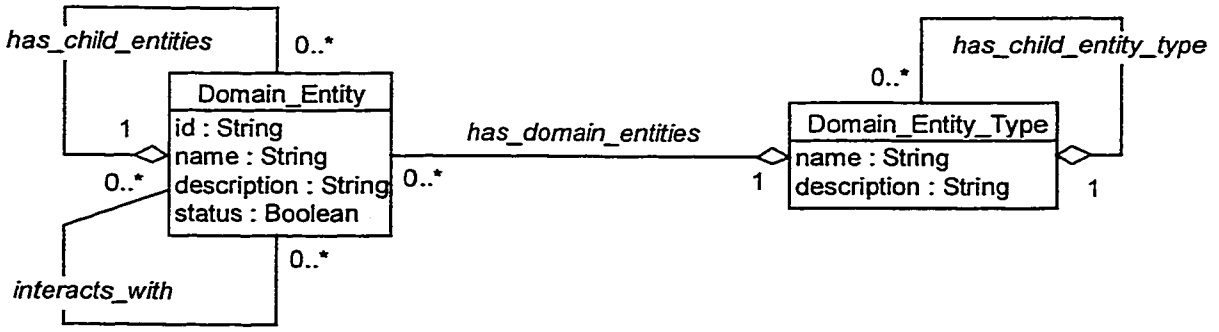


Figure 5.4: Representation of domain entity types

In order to connect domain entities with their appropriate domain entity types, a one-to-many bi-directional aggregation relationship called "has_domain_entities" has been introduced between the "Domain_Entity_Type" class and the "Domain_Entity" class. This relationship restricts every domain entity to pertain to one and only one domain entity type. In order to facilitate the representation of complex domains, a one-to-many bi-directional reflexive aggregation relationship called "has_child_entity_type" has been introduced in the "Domain_Entity_Type" class. This relationship not only allows the hierarchical decomposition of cases but also controls this decomposition and guarantees the generation of logical and meaningful case structures. This means that domain entities that belong to a parent domain entity type cannot be decomposed into any other domain entities except those that belong to child domain entity types. For example, a foundation

system cannot be decomposed directly into piles and pile caps, but has to be decomposed first into foundation assemblies that can be decomposed further into piles and pile caps. This is because the "Foundation System" domain entity type is a parent of the "Foundation Assembly" domain entity type and not of the "Pile" or "Pile Cap" domain entity types (refer to the conceptual model shown in Figures 4.12 and 4.13).

5.3.3 ATTRIBUTES AND THEIR VALUES

The natural place for storing data in the standard object-oriented representational paradigm is object data members that are called object attributes. These attributes are static components; once they are defined by the system developer at the implementation sub-phase, they cannot be changed later by the system user at run-time (Messner et al. 1994). Moreover, data types of these attributes must also be determined at the implementation sub-phase and cannot be altered afterwards. Any attempt of changing an object structure after its implementation by adding new attributes, removing existing ones, or changing attribute data types may result in losing object data (Object Design 1998b). The rigidity of the static object structure and the risk of losing object data due to any changes in this structure do not fulfill the extensibility and data evolution requirements of infrastructure management domains. These requirements necessitate the use of another approach in storing object data. In the proposed approach, the storage of object data has been shifted from inside the object (using object data members) to outside the object by using other objects instantiated from the two classes: "Attribute" and "Attribute_Value" shown in Figure 5.5.

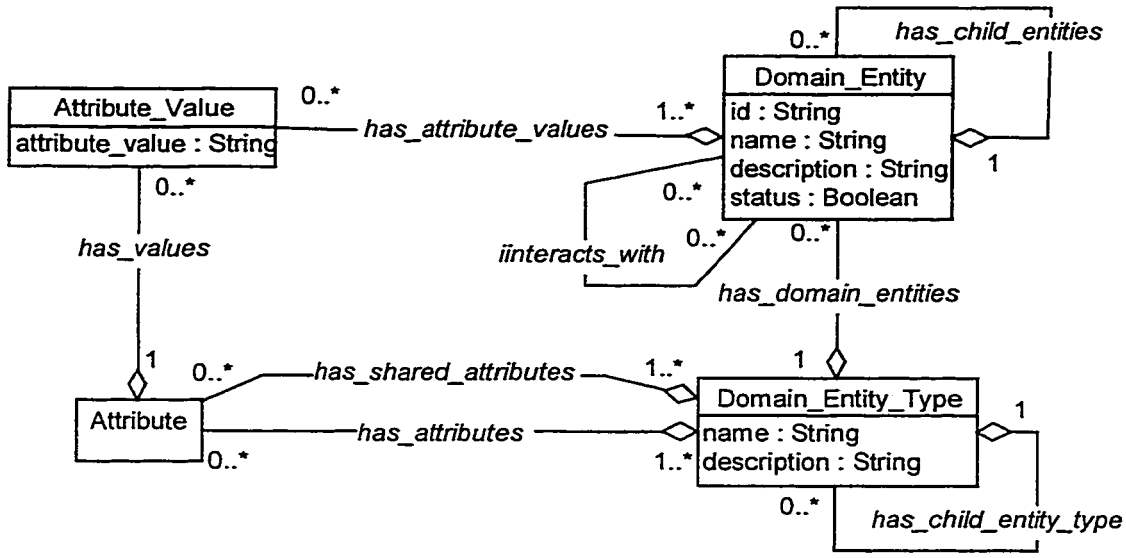


Figure 5.5: Representation of attributes and their values

Objects instantiated from the "Attribute" class represent the attributes of a specific domain entity type. Each attribute contains many data members that describe different attribute properties, such as attribute name, description, default value, weight, role, etc. These properties will be discussed in detail in the "Case Retrieval" section. The "Domain_Entity_Type" class has a many-to-many bi-directional aggregation relationship with the "Attribute" class called "has_attributes". This type of relationship allows different domain entity types to reuse the same attribute, which means defining the attribute once and utilizing it in describing several concepts. For example, the structural system attribute may be reused by the girder, slab, and truss domain entity types. The "Domain_Entity_Type" class has another many-to-many bi-directional aggregation relationship with the "Attribute" called "has_shared_attributes". This type of relationship allows different domain entity types to reuse the same attributes and to share the attribute values among their domain entities, which means defining the attribute values once and utilizing them in describing several domain entities. For example, the value of the bridge region attribute can be shared by all of the domain entities that constitute this bridge.

In the class diagram shown above, the data of domain entities are stored neither in the domain entity objects nor in their attribute objects. Separate objects instantiated from the "Attribute_Value" class are used to store these data. Each attribute object has a one-to-many bi-directional aggregation relationship, called "has_values", with attribute value objects that belong to that attribute. Each of these attribute value objects corresponds to different domain entity objects through the many-to-many bi-directional aggregation relationship called "has_attribute_values". This type of relationship allows an attribute value object to be shared among many domain entities. Each attribute value object has a string data member called "attribute_value" that stores the actual values and supports different types of data, such as String, Boolean, Integer, and Float. Figure 5.6 shows a schematic representation of the proposed approach using an example of a bridge domain entity type that has two domain entities and three attributes.

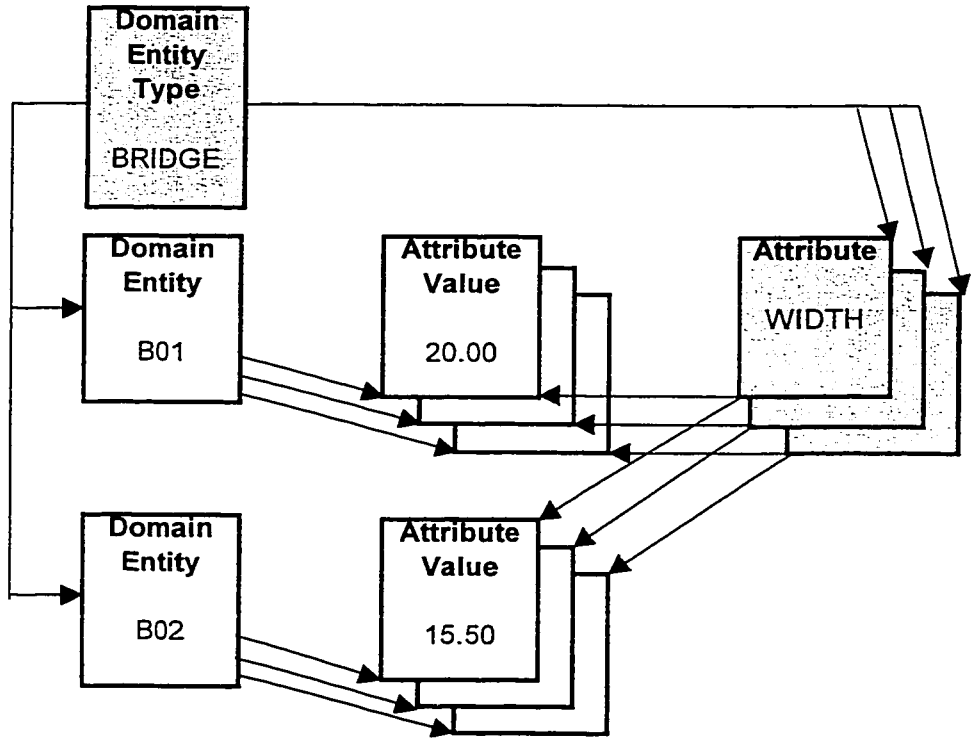


Figure 5.6: Schematic representation of the proposed approach in storing entity data

There are six advantages for using this approach to storing domain entity data. First, domain entity types are flexible enough to allow the addition, removal, and update of their attributes at run-time without affecting their object structures. Second, having separate objects for attributes facilitates the storage of retrieval knowledge inside these objects. Third, representing attributes distinct to domain entity types allows reusing these attributes among several domain entity types. Fourth, having separate objects for attribute values facilitates modifying attribute objects with no risk of losing these values. Fifth, representing attribute values in a string data type allows storing any other data type and changing it at run-time. Sixth, representing attribute values distinct to domain entities allows sharing these values among several different domain. This approach has great potential in modeling domains that are not well defined and whose attributes are unpredictable and subjected to possible changes.

5.3.4 TIME-DEPENDENT SETS AND THEIR TYPES

In infrastructure management domains, some concepts are dynamic in nature (i.e. contain time-dependent data), such as bridge inspection, bridge maintenance, and element condition. To represent these concepts in the domain model, new containers, different from domain entities, are introduced into the design model. These containers are objects instantiated from a class called "Time-Dependent Set" as shown in Figure 5.7. Each object encapsulates the values of several time-dependent attributes that correspond to a specific value of another attribute called time-attribute. Example is the time-dependent set that stores the values of the MCR and PCR of a bridge deck at specific age. A many-to-many bi-directional aggregation relationship called "has_attribute_values" has been

introduced between the "Time_Dependent_Set" class and the "Attribute_Value" class to allow sharing of attribute values among different time-dependent sets. These sets cannot be stand-alone objects; they have to be contained in at least one domain entity. A many-to-many bi-directional aggregation relationship called "has_time_dependent_sets" has been introduced between "Domain_Entity" class and "Time_Dependent_Set" class to define which time dependent sets belong to which domain entities. This type of relationship allows the sharing of time-dependent sets among different domain entities. Each time-dependent set has two data members: "id" to uniquely identify the set and "description" to describe the set in details. A reflexive bi-directional one-to-many aggregation relationship called "has_child_sets" has been introduced among time-dependent sets to allow the aggregation of many time-dependent sets into a parent set. For instance, bridge inspections may contain several element inspections and both of them are represented as time-dependent sets.

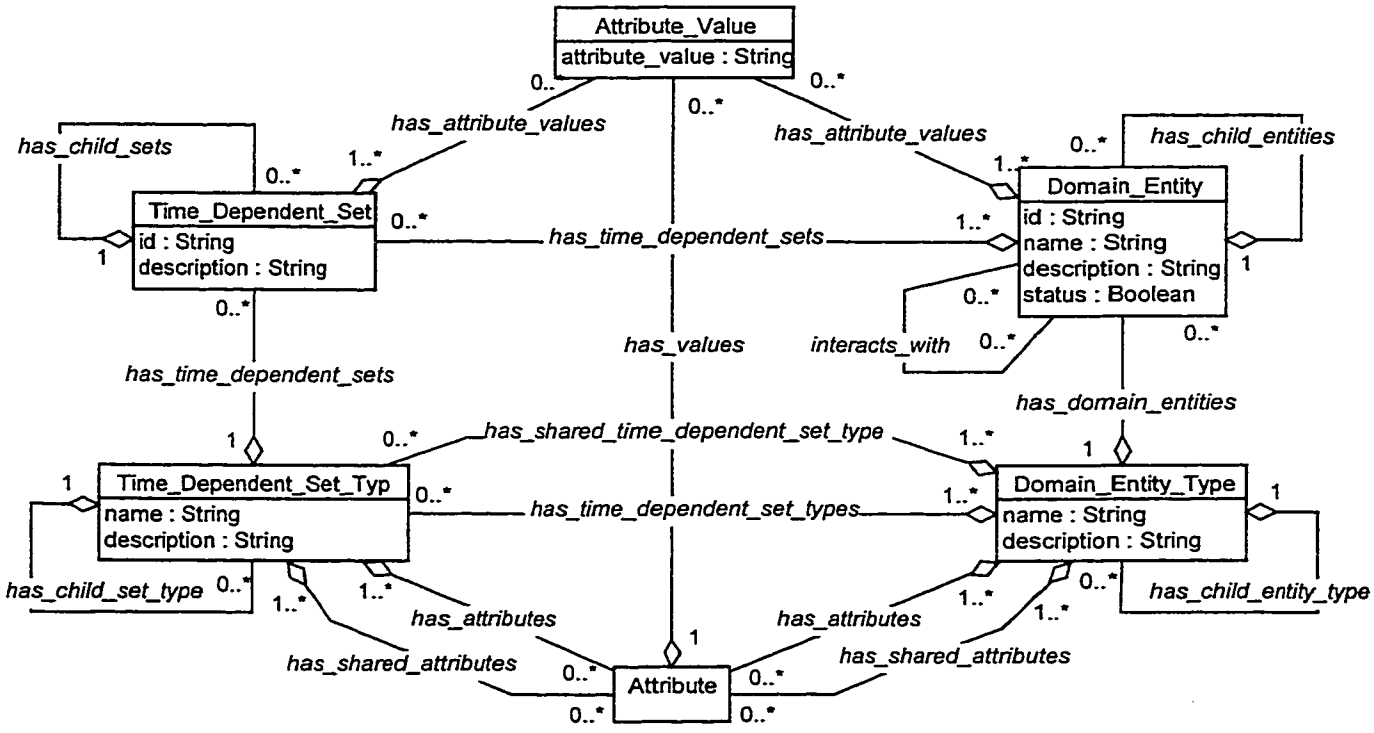


Figure 5.7: Representation of time-dependent sets and their types

As with domain entities, time-dependent sets that have common features and common types of relationships are grouped and represented as objects instantiated from a class called "Time_Dependent_Set_Type" as shown in Figure 5.7. Time-dependent set types cannot be stand-alone objects but have to be contained in at least one domain entity type. Each time-dependent set type has two data members: "name" to uniquely identifies the set type and "description" to describe it in details. Two many-to-many bi-directional aggregation relationships called "has_time_dependent_set_types" and "has_shared_time_dependent_set_types" have been introduced between the "Domain_Entity_Type" and "Time_Dependent_Set_Type" classes to allow, respectively, reusing and sharing time-dependent set types among different domain entity types. Attributes are defined for time-dependent set types the same way they are defined for domain entity types. A reflexive one-to-many bi-directional aggregation relationship called "has_child_set_type" has been introduced among time-dependent-set-types to represent the aggregation of some time dependent set types into a parent set type.

Figure 5.8 shows an example that illustrates the case representation aspect in the developed CBR shell. The figure shows two domain entities, beam "B20" on the upper-left and column "C21" on the upper-right, that belong to the domain entity type "Bridge_Element" shown at the top. The entity type has two attributes: "Material" and "Shape" and a time-dependent set type "Element_Condition" that has two attributes: "Age" and "Condition". The two domain entities store their static data in two "Attribute_Value" objects and their dynamic data in two "Time_Dependent_Set" objects and their respective "Attribute_Value" objects.

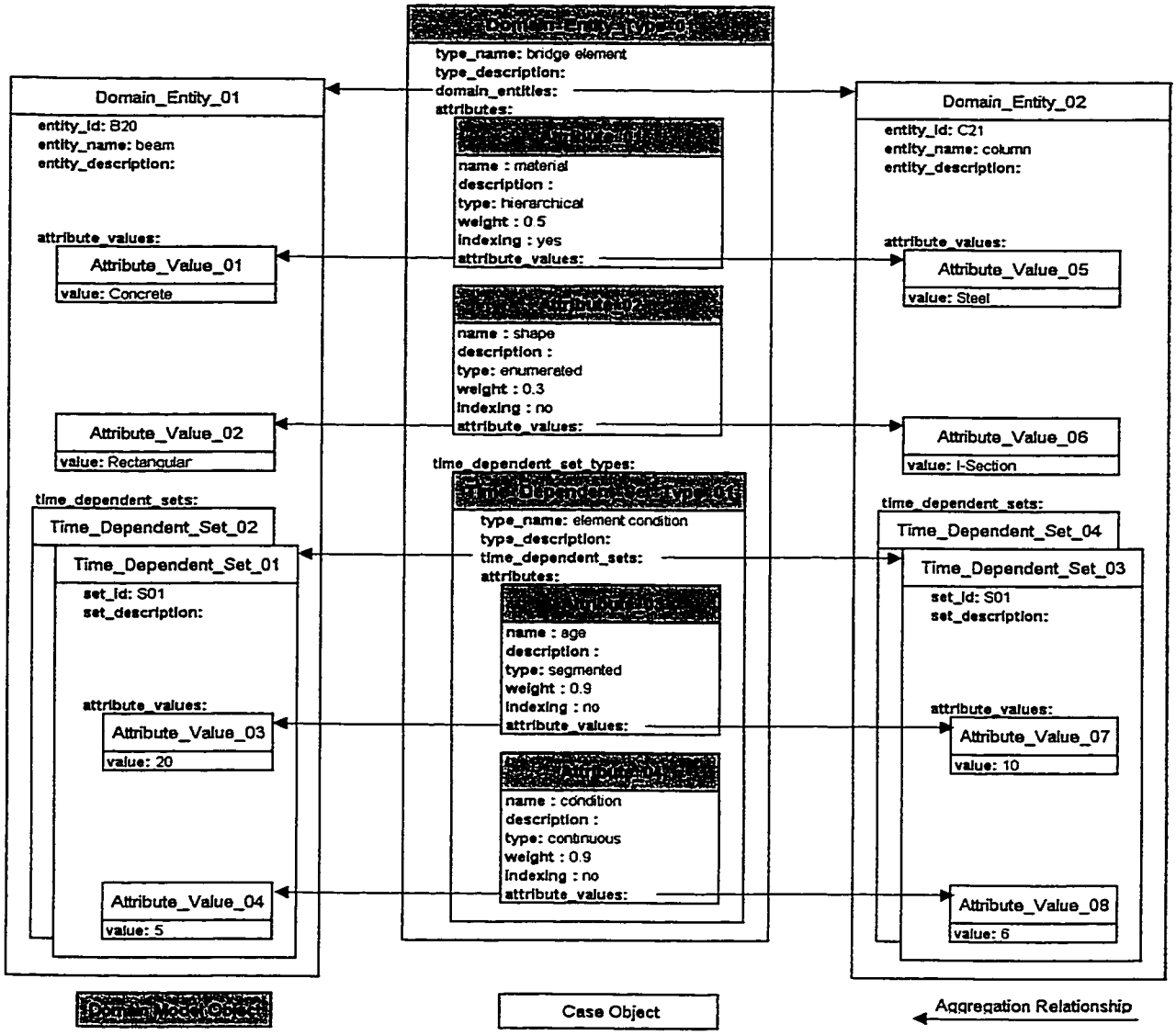


Figure 5.8: Example of case representation

Domain entity types and time-dependent set types are the building blocks of the domain model. Defining these entity types, set types, and their relationships is the first step in developing a domain model. Some rules guide these definitions:

1. Only one root domain entity type is allowed in any domain model.
2. None of the child domain entity types can have two parents.
3. Although many root time-dependent set types are allowed, none of the time-dependent set types can belong to two parents.

4. Each time-dependent set type must belong to at least one domain entity type.
5. Time-dependent set types can be either reused or shared but not both. This is because reusing a time-dependent set type leads to having different time-dependent sets that belong to different domain entities. This would be confusing to the system as which of these sets would be shared.

5.3.5 CASE TEMPLATES

Although bridges are rarely identical, they can be categorized into groups called bridge types. Each bridge type maintains the same hierarchical decomposition (case structure) for all bridges that belong to that type. Bridges of the same bridge type may differ in the number of bridge components, but this difference does not violate the similarity in their decomposition. For aggregating the condition of different bridge components to obtain the overall bridge condition, every bridge type has a set of numeric values that represent the relative importance of its components, which are called entity weights. These values correspond to the balance factors that are used in the BMS of Quebec (MTQ 1997). Every bridge type is characterized also by different interaction mechanisms among its components. The participation of different bridge components in these mechanisms differs from one bridge type to another.

In order to gain the benefit of having bridge types, the concept of case templates is used in the developed CBR shell. Each case template represents one infrastructure facility type (e.g. bridge type) that has a particular case structure, balance factors, and entities interaction. By using these templates, new cases can be created quickly and with less

possibility of errors. Case templates are represented in the design model as a collection of domain entities (i.e. the same as cases are represented). Three data members have been introduced into the "Domain_Entity" class as shown in Figure 5.9 to distinguish cases from case templates, to store balance factors for every case template, and to define the participation of domain entities in different interaction mechanisms. The "template_case" data member is set to TRUE only if the domain entity is used in a case template; otherwise, it is set to FALSE. The "weight" data member stores the balance factor of the domain entity, which represents the relative importance of that domain entity in the bridge type. The "interaction_weight" data member represents the participation of the domain entity in different interaction mechanisms.

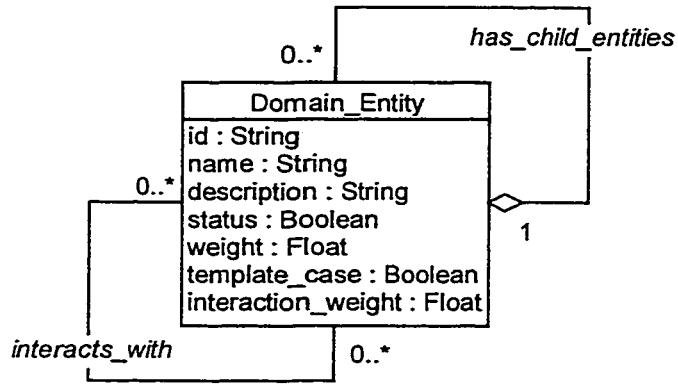


Figure 5.9: Complete representation of domain entities

When a case template is selected to generate new cases in the case library, new domain entities that are identical to those constituting the case template are created. These domain entities have also relationships among them, which represent the case structure and entities interactions, exactly the same as in the case template. Only the "template_case" data member is set to FALSE for all the generated domain entities.

5.3.6 LOCATION IDENTIFIERS

Although the hierarchical decomposition of infrastructure facilities such as bridges is a powerful way for representing their components, this representation may be disadvantageous if it cannot be translated into currently existing representations. Most of the current representations decompose infrastructure facilities into components according to the location of these components. For example, bridges are usually decomposed into axes and spans that have names to identify the components located on them (see Figure 4.3). Therefore, the system should be capable of storing location identifiers in every case and connecting them with their corresponding domain entities in the hierarchical decomposition.

Location identifiers are represented as objects instantiated from a class called "Location_Identifier". Each location identifier has three data members as shown in Figure 5.10: "type", "name", and "description". The "type" data member determines the category of the identifier such as an axis, a span, a section, etc. The "name" data member is a unique label that distinguishes the identifier from others in the same category. Examples are the axes A01, A02, and A03 and the spans S01 and S02 of the two-span bridge shown in Figure 5.11. The "description" data member stores any additional information about that identifier. Location identifiers from the same or different categories need to be connected. Figure 5.11 shows an example for the location identifier S01 that is connected with location identifiers A01 and A02 from a different category. As an example for connecting location identifiers from the same category, the connections among the neighboring sections of a highway. To represent this connection, a reflexive

bi-directional many-to-many association relationship called "connected_to" has been introduced into the "Location_Identifier" class.

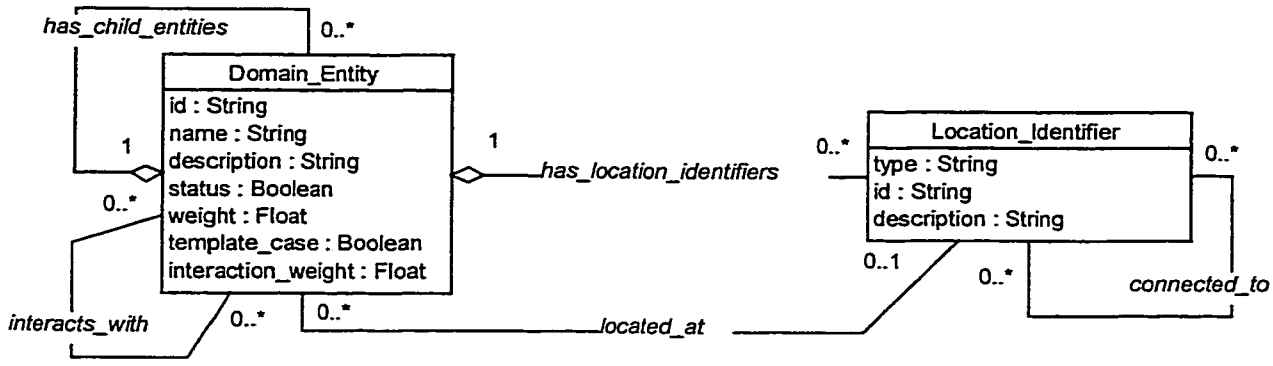
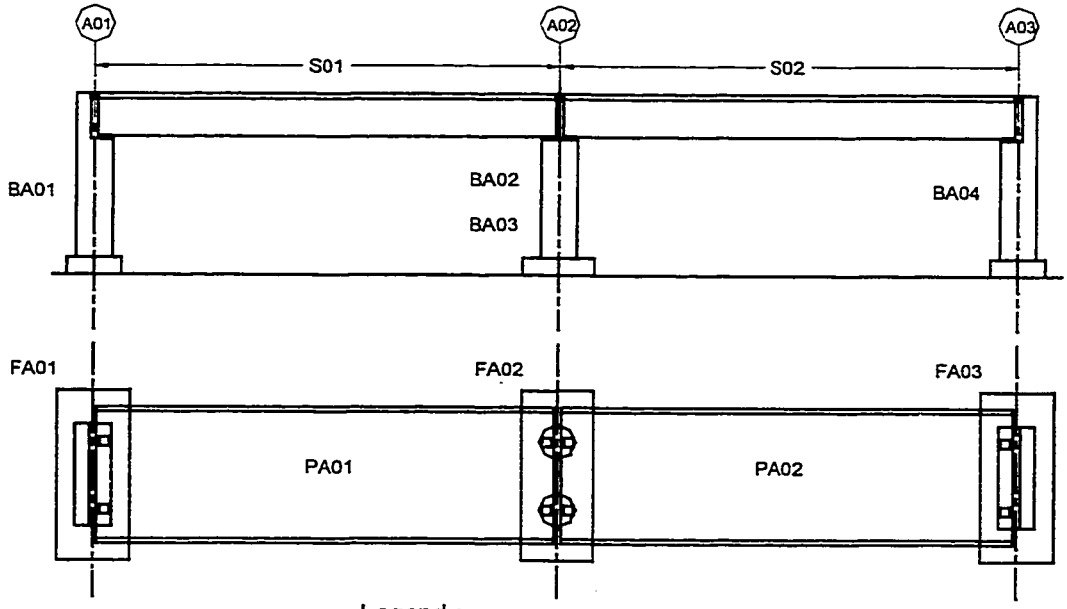


Figure 5.10: Representation of location identifiers and their relationships



Legend :

A Axis
S Span
FA Foundation Assembly
BA Substructure Assembly
PA Superstructure Assembly

Figure 5.11: Example of location identifiers

Two other types of relationship have been introduced between the "Domain_Entity" class and the "Location_Identifier" class. First is a one-to-many bi-directional aggregation relationship called "has_location_identifiers". This type of relationship is used to define the location identifiers that belong to a specific infrastructure facility. Therefore, this type of relationship is established between the location identifiers and the root domain entity, which represents the infrastructure facility. Second is a many-to-one bi-directional association relationship called "located_at". This type of relationship is used to connect each domain entity with the location identifier, where the domain entity is located at. Figure 5.12 shows the hierarchical decomposition of the bridge shown in Figure 5.11 to the level of bridge assemblies along with the location identifiers used for these assemblies and their connections.

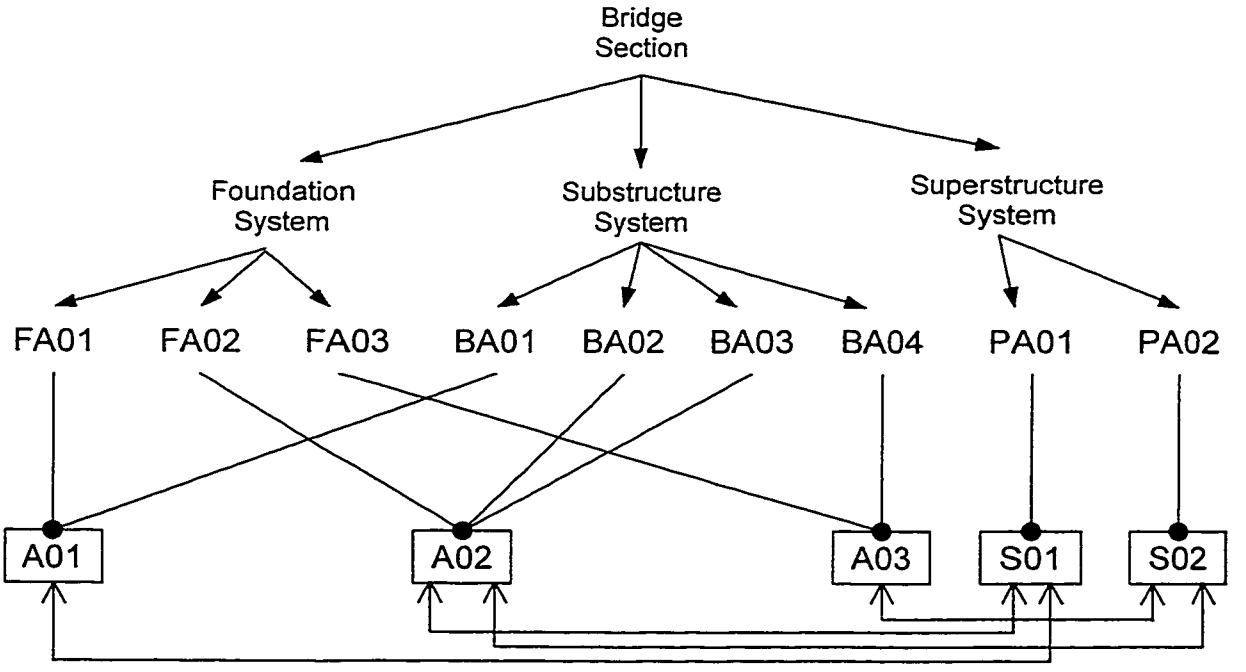


Figure 5.12: Location identifiers connecting bridge assemblies and their relationships

5.4 CASE RETRIEVAL

Case retrieval is a search and match process. The case-based reasoner starts by searching the whole case library for cases that are similar to the query case defined by the user. This search often results in too many cases that need to be examined in order to determine the "best case". The term "best case" means the case that is the most similar to the query one..This similarity is evaluated during the matching process. There are three approaches described in the literature for case matching (Maher et. al. 1995):

1- **Maximum number of matched features:** This approach considers the best case as the case that has the highest number of matched features. Features are the attributes that describe the case contents (Kolodner 1993). The comparison of corresponding attribute values employs similarity measure techniques that allow partial matching. This approach is useful only when attributes have the same importance.

2- **Nearest-neighbor matching:** This approach is similar to the previous one except it accounts for the differences in the importance of attributes and allows the importance to be represented as crisp numbers (weights) and as fuzzy numbers (levels of importance). The best case is then determined as the one that has the highest weighted average of attribute similarities.

3- **Context-dependent matching:** This approach attempts to figure out the best case based on domain specific knowledge. The key to this approach is that the assessment of the degree of similarity and features importance are not static but are highly dependent on the context of the current problem. In Stanfill (1988) a method that can dynamically change the importance of the features by analyzing the context of the current problem and applying domain-specific knowledge models is presented.

Among the three above-mentioned approaches of case matching, the nearest-neighbor approach has been selected for matching cases in the developed system. This is because nearest-neighbor offers a level of precision that is in between the other approaches. Nearest-neighbor recognizes the differences in the importance of features with no need for domain-specific knowledge which may be difficult to acquire. It allows the representation of attribute weights and degrees of similarity as fuzzy numbers. It also facilitates the calculation of global similarity in a systematic manner that is easy to implement.

In order to carry on both search and match processes, knowledge about every attribute in terms of weight, role, type, possible values, similarities among these values, and default value is indispensable. This knowledge is called retrieval knowledge. The following sections discuss in detail how this knowledge is represented in the shell design model and how the retrieval algorithm manipulates this knowledge to select the best case.

5.4.1 REPRESENTATION OF RETRIEVAL KNOWLEDGE

According to the design model developed in the previous section, attributes are represented as separate objects that are contained in either domain entity types or time-dependent set types (refer to section 5.3.3). These objects are the most appropriate place for storing retrieval knowledge. Therefore, several data members have been introduced in the "Attribute" class, as shown in Figure 5.13, to represent this knowledge. These data members are explained below.

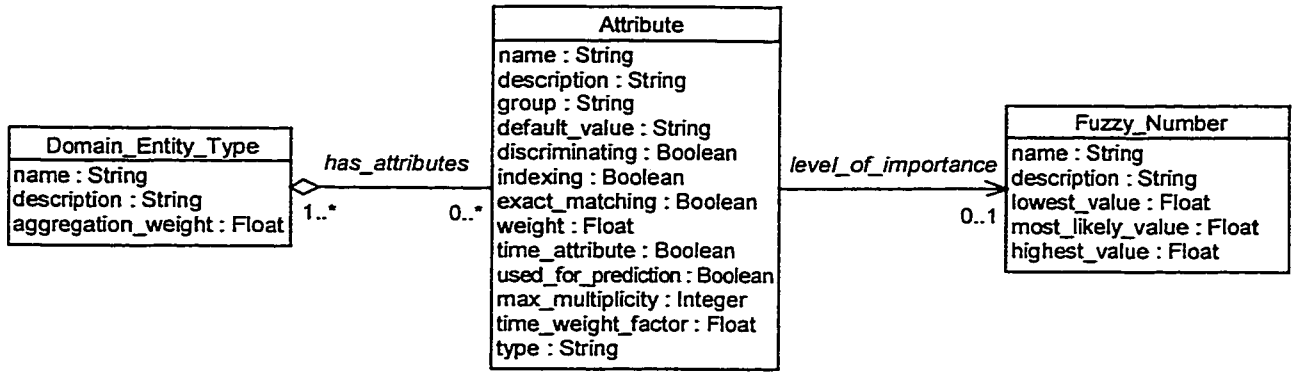


Figure 5.13: Representation of attributes

The "name" and "description" data members are, respectively, a non-null unique identifier of an attribute and a narrative description of that attribute. The "group" data member provides a way to gather attributes that are related and belong to the same feature. For instance, attributes such as material type, material strength, and covering material can be grouped under one title called "Element Material". Grouping attributes does not affect the retrieval process but provides an efficient way to present attributes to the user. The "default_value" data member stores an attribute value that is used whenever there is no value defined for that attribute. It is necessary for an attribute to have a value, either default or defined, in order to be considered in the matching process.

The attribute role in the retrieval process is an indispensable part of the retrieval knowledge. Attributes may be discriminating, indexing, exact matching or non-discriminating. A discriminating attribute is an attribute that differentiates between cases according to its values, such as the deck structural system attribute that differentiates between simply supported decks and continuous decks. An indexing attribute is a discriminating attribute that filters out the cases that do not have a full similarity of values with the query, such as the deck material attribute that filters out the cases with

wooden decks when the query case has a concrete deck. An exact matching attribute is an indexing attribute that filters out the cases that do not have identical values for that attribute as the query case, such as the deck MCR attribute that filters out the cases with deck MCR equal to 4.9 when the query case has a deck MCR equal to 5. A non-discriminating attribute is an annotation attribute that does not have any function in the retrieval process, such as a bridge name. Attribute role is determined by using the three Boolean data members: "discriminating", "indexing", and "exact_matching".

The "weight" data member indicates how much attention should be paid to the attribute during the matching process (Kolodner 1993). It reflects the importance of that attribute relative to other case attributes. This data member has a float type and can take a value between 0 and 1 where 0 represents a negligible attribute while 1 represents an extremely important one. One method for obtaining attribute weights requires domain experts to assign numeric values that represent the absolute importance of case attributes. This is, of course, not an easy task because crisp numeric values do not reflect the way domain experts recognize attribute importance. This problem has been solved by introducing fuzzy numbers that are equivalent to attribute weights and consider the uncertainty of human judgement (Tee et al. 1988). Each fuzzy number is represented as an object instantiated from a class called "Fuzzy_Number" as shown in Figure 5.13. This object has a name, description, and three float numeric values that represent the values of a triangular fuzzy number (TFN): the lowest value, the most likely value, and the highest value. The relationship called "level_of_importance" connects the attribute object with the fuzzy number object that assesses its importance. Examples of the fuzzy numbers

defined to assess attribute importance are shown in Figure 5.14. These fuzzy numbers capture the uncertainty of human judgement and recognize relative importance among case attributes. Another method for obtaining attribute weights requires the statistical analysis of a set of cases. This analysis results in coefficients of correlation between different case attributes and the case solution. These coefficients represent the contribution of different attributes in predicting the solution (Kolodner 1993). Similarly, domain entity type objects contain a data member called "aggregation_weight" that is considered as a part of retrieval knowledge. This data member represents the importance of sub-cases relative to the main case in the matching process (this will be discussed in section 5.4.2 in details).

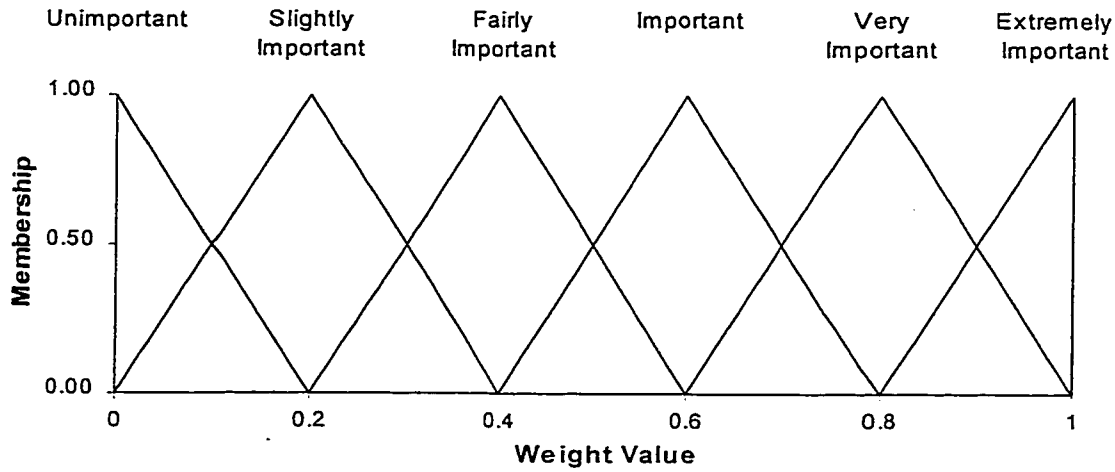


Figure 5.14: Fuzzy numbers representing attribute levels of importance

The retrieval process needs more knowledge for the attributes defined for time-dependent set types. Two Boolean data members are defined for these attributes: "time_attribute" and "used_for_prediction". Only one attribute in every time-dependent set type must be selected as a time attribute. Such an attribute stores the time point at which all the data of the time-dependent set is recorded. For example, the maintenance

date is the time attribute of the "Bridge_Maintenance" time-dependent set. For every domain entity type, only one attribute from its time-dependent set types can be used for prediction, so the system can recognize the values that need to be predicted. This attribute is the one whose future values are predicted by the system. For example, the MCR is the attribute used for prediction in the "Deck" domain entity type. Two additional data members are needed for such an attribute: "max_multiplicity" and "time_weight_factors". The "max_multiplicity" determines the number of points sufficient for matching the values of the attribute used for prediction. The "time_weight_factor" determines how attribute weight declines with time. Details about these data members will be discussed later with the retrieval algorithm in section 5.4.2.

Attributes, in general, can be categorized into three types: numeric, symbolic, and text. Numeric attributes are those attributes whose values can be represented on a quantitative scale. Examples of these attributes are traffic volume, age, construction date, span, etc. The value of a numeric attribute can be an integer, float, or date. Symbolic attributes are those attributes whose values can be represented on a qualitative scale. Examples of these attributes are element material, structural system, bridge region, highway classification, etc. Text attributes are those attributes whose values cannot be represented on either a quantitative or qualitative scale. These attributes are used for annotations such as bridge address or contractor name. Text attributes do not participate in the retrieval process and, therefore, do not have retrieval knowledge. Numeric and symbolic attributes have retrieval knowledge that is stored in two classes inherited from the "Attribute" class, "Numeric_Attribute" and "Symbolic_Attribute", and described below.

5.4.1.1 Numeric Attributes

Every numeric attribute has the data members defined earlier for the "Attribute" class in addition to the data members defined for the "Numeric_Attribute" class shown in Figure 5.15. The "unit", "upper_limit", and "lower_limit" data members are used to maintain correct interpretations and meaningful entries for the values of numeric attribute. In order to distinguish between an integer or float attribute and a date one, a Boolean data member called "date_attribute" is used. The reason for this differentiation is that mathematical operations are defined differently for dates. In some numeric attributes, two (or more) attribute values are related. Some attribute values can be derived from the values of their related attribute (Elmasri and Navathe 1994). For example, the "Inspection Age" of a bridge, its "Construction Date", and its "Inspection Date" are related attributes. The value of the "Inspection Age" attribute can be calculated as the difference between the values of "Inspection Date" and "Construction Date" attributes. Hence, the "Inspection Age" is called a derived attribute, while the "Inspection Date", and the "Construction Date" are called stored attributes.

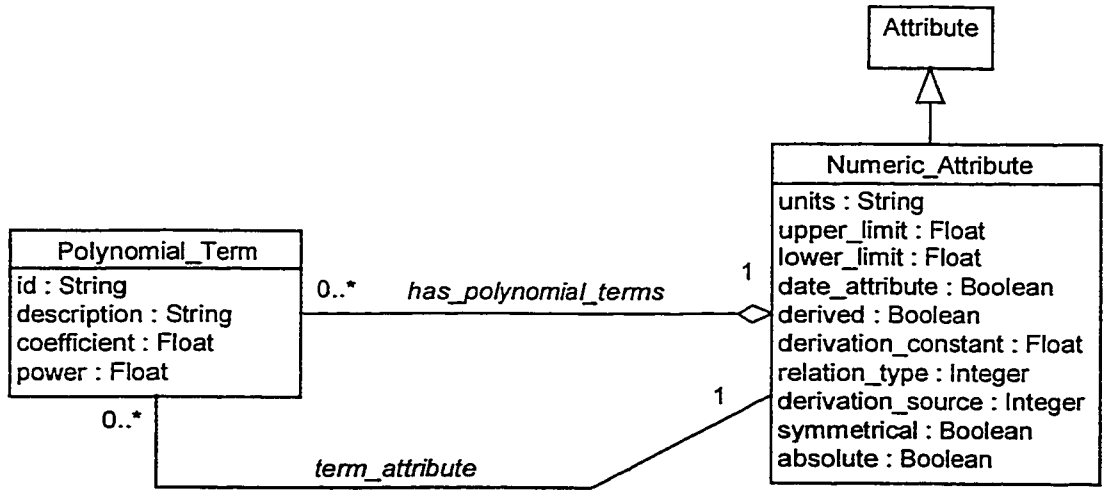


Figure 5.15: Representation of derivation equations

Whenever there is a derived attribute, a derivation equation has to be built to calculate its value. This derivation equation is represented as a polynomial that has a collection of polynomial terms and a constant. Each polynomial term is represented as an object instantiated from a class called "Polynomial_Term" that has four data members: "id" to uniquely identify the term, "description" to store additional information about the term, and "coefficient" and "power" to store float values used in calculating the term value. A one-to-many bi-directional aggregation relationship called "has_polynomial_terms" has been introduced between the "Polynomial_Term" class and the "Numeric_Attribute" class to determine the polynomial terms used in building the derivation equation. A many-to-one bi-directional association relationship called "term_attribute" has been introduced between the "Polynomial_Term" class and the "Numeric_Attribute" class to determine the numeric attribute whose value is used to calculate the polynomial term. The "Numeric_Attribute" data members, "derivation_constant" and "relation_type", are used to calculate the total value of the derived attribute according to the type of relation between terms (summation, multiplication, maximization, or minimization).

Not all numeric attributes defined for a case can be used as term attributes in deriving another numeric attribute in that case. There should be a unique path between the container of a derived attribute and the containers of its term attributes. In general, the value of an attribute in any container (domain entity or time-dependent set) can be derived from the values of attributes that belong to the same container or to one of its parent containers. In case of time-dependent sets, an attribute value can be derived from the values of attributes that belong to the parent domain entity. Infrastructure

management domains require aggregating the condition of child entities in order to obtain the condition of the parent entity and aggregating the condition of interacting entities in order to consider their effects. Therefore, a data member called "derivation_source" is introduced into the "Numeric_Attribute" class in order to determine if the source of derivation is the same or parent container, child entities, or interacting entities.

For the purpose of measuring the similarity between values of a numeric attribute, two types of numeric attributes are defined: continuous attribute and segmented attribute. These types are represented as two classes inherited from the base class "Numeric_Attribute": "Continuous_Atribute" class and "Segmented_Attribute" class. Below is the definition of each type along with the description of its similarity measure.

Continuous Attributes

A continuous attribute is a numeric attribute that has a semi-continuous degree of similarity that depends on the difference between the attribute values. A similarity function is used to calculate the degree of similarity between two attribute values as a function of the difference between them; the larger the difference between the values the lower their degree of similarity (Kolodner 1993). The degree of similarity S of a continuous attribute is computed using the similarity function $f(D)$ as follows:

$$S = f(D) \qquad \text{where, } D = Vc - Vq$$

$$f(D) = \begin{cases} 0 & \text{when} \quad |D| \geq I \\ \left(\frac{I - |D|}{I}\right)^R & \text{when} \quad I \geq |D| \geq 0 \end{cases}$$

where,

Vc	Value of the continuous attribute in the case
Vq	Value of the continuous attribute in the query
I	Interval of the difference D in which the similarity is greater than 0
R	Rate of change of the similarity S with the difference D

The similarity function $f(D)$ is designed to provide a complete similarity ($S = 1$) when D equals 0 and to provide a complete dissimilarity ($S = 0$) when D equals or exceeds a certain interval I specified by the user. The unsigned value of D is used to make the similarity function symmetrical; that means there is no difference between $Vc > Vq$ and $Vq > Vc$ as long as D is the same. To use the similarity function with various continuous attributes and provide the user with a customized function pattern, a parameter R is added. This parameter represents the rate by which the degree of similarity changes from 1 to 0 while D increases from 0 to I. Figure 5.16 shows different patterns of similarity functions with various values of R.

Example: Assume that the traffic volume on a query bridge Vq = 10000 pcu (passenger car unit per day), and the traffic volume on a case bridge Vc = 8000 pcu, then D = 8000 – 10000 = -2000 pcu. If the user has set the interval of the maximum allowable difference I = 3000 pcu, and the function pattern parameter R = 1, then the degree of similarity S will be calculated as follows:

$$S = \left(\frac{3000 - |-2000|}{3000} \right)^1 = 0.33$$

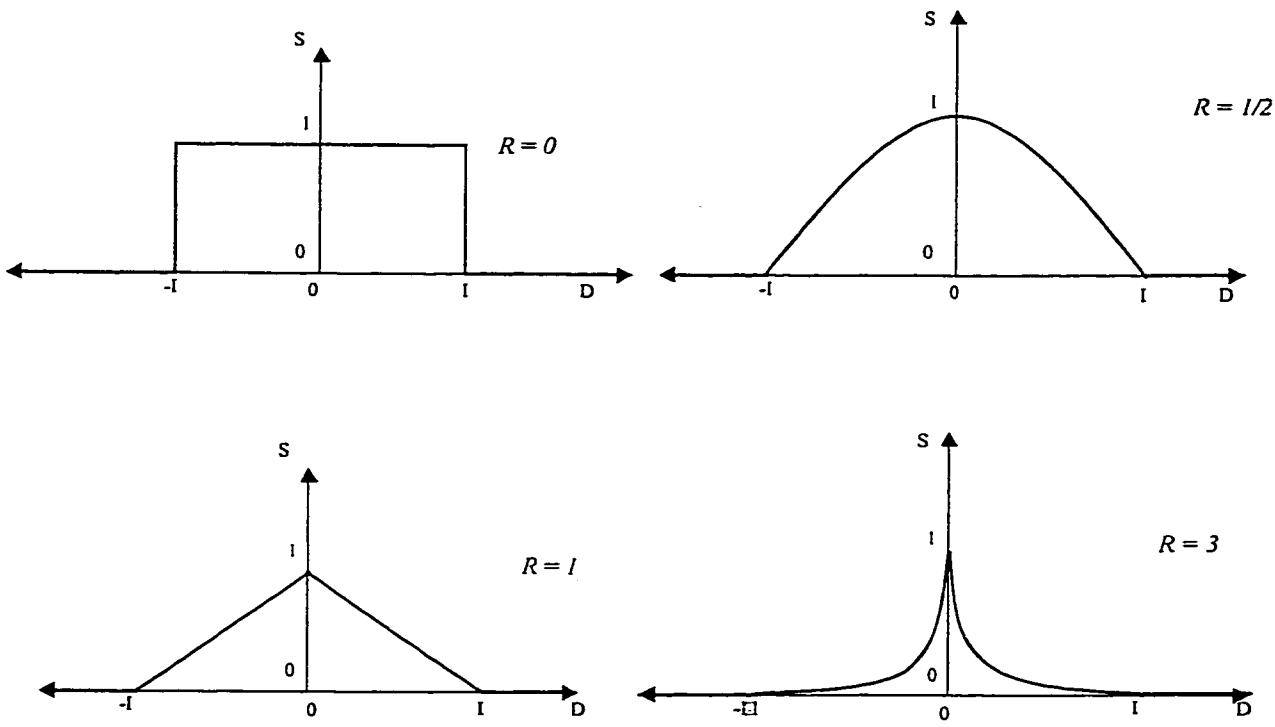


Figure 5.16: Patterns of similarity function

This method for calculating the degree of similarrity for a continuous attribute can be extended by allowing the similarity function to be asymmetrical. Such a function differentiates between having $Vc > Vq$ and $Vq > Vc$ while the difference is the same. This extension is done by defining two values for both the interval I and the parameter P: one value when D is positive (the right side of the curve I_R, and P_R), and the other value when D is negative (the left side of the curve I_L, and P_L). This similarity function can be written as follows:

$$f(D) = \begin{cases} 0 & when \quad D \geq I_R \\ \left(\frac{I_R - D}{I_R}\right)^{R_R} & when \quad I_R \geq D \geq 0 \\ \left(\frac{I_L + D}{I_L}\right)^{R_L} & when \quad -I_L \leq D \leq 0 \\ 0 & when \quad D \leq -I_L \end{cases}$$

This method can also be extended by evaluating D as a relative value rather than an absolute one. This is done by calculating D as the difference between Vc and Vq and dividing the result by Vq. In this situation, user inputs for both I_R and I_L should also be relative values. For instance, in the previous example the relative value of the interval I is 3000/10000 = 0.3 and of the difference D is –2000/10000 = -0.2. Two Boolean data members, "symmetrical" and "absolute", have been introduced into the "Numeric_Attribute" class in order to determine, respectively, if the similarity function is symmetrical or not and if the calculations are based on relative or absolute values.

Segmented Attributes

A segmented attribute is a numeric attribute that has discrete degrees of similarity S for different ranges (segments) of the difference D between the attribute values. Similarity ranges are used to define the limits of each range and the corresponding degree of similarity. This technique is used when small differences in attribute values are irrelevant to the degree of similarity (Dzeng and Tommelein 1995). As an example, assume that a bridge deck age ranges from 0 to 60 years. This means that D may take any value from –60 to +60. Table 5.1 shows six similarity ranges defined for that attribute.

Segment	Range of D	Degree of similarity S	Level of similarity
S_1	$-60 \le D < -8$	0	Dissimilar
S_2	$-8 \le D < -3$	0.4	Fairly similar
S_3	$-3 \le D < 2$	1	Extremely Similar
S_4	$2 \le D < 6$	0.6	Similar
S_5	$6 \le D < 12$	0.2	Slightly similar
S_6	$12 \le D < 60$	0	Dissimilar

Table 5.1: Similarity ranges of the deck age attribute

Similarity ranges provide a greater flexibility over the similarity function in determining the degree of similarity because there is no limit for the number of ranges that can be specified by the user. Also, similarity ranges can incorporate the representation of the levels of similarity as fuzzy numbers as shown in Figure 5.17. This representation is more realistic because it simulates the way the domain experts express similarity. In addition, similarity ranges do not restrict the similarity calculation to a single continuous function over a big interval of D. They resemble multiple similarity functions with constant rate ($P = 0$) and variable interval I. Figure 5.18 represents this analogy by depicting the values shown in Table 5.1.

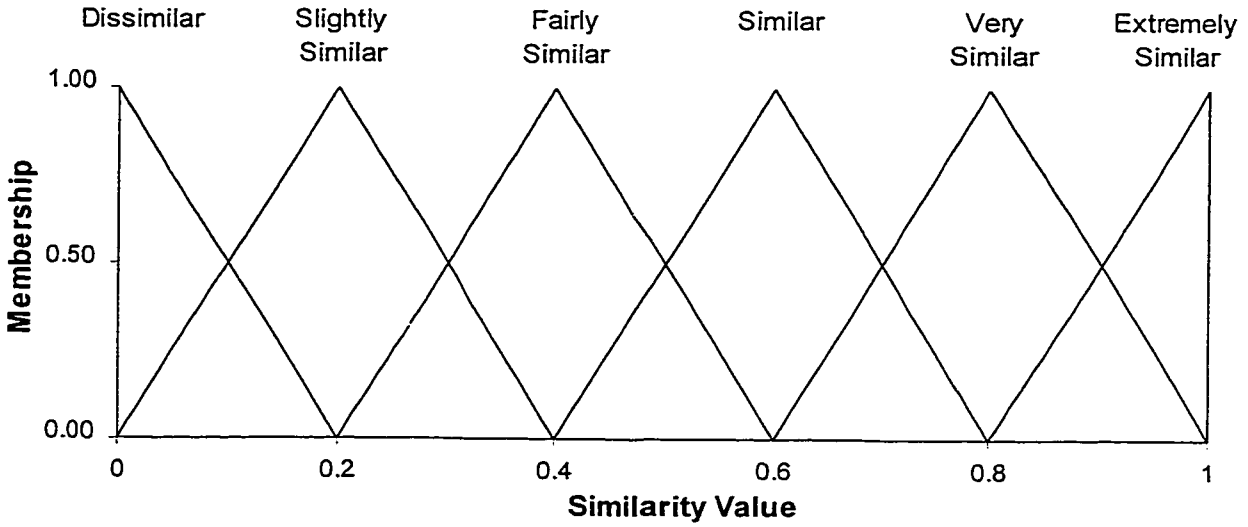


Figure 5.17: Fuzzy numbers representing levels of similarity

Figure 5.19 shows a part of the CBR shell design model that represents the retrieval knowledge of numeric attributes along with the methods used for measuring their similarities.

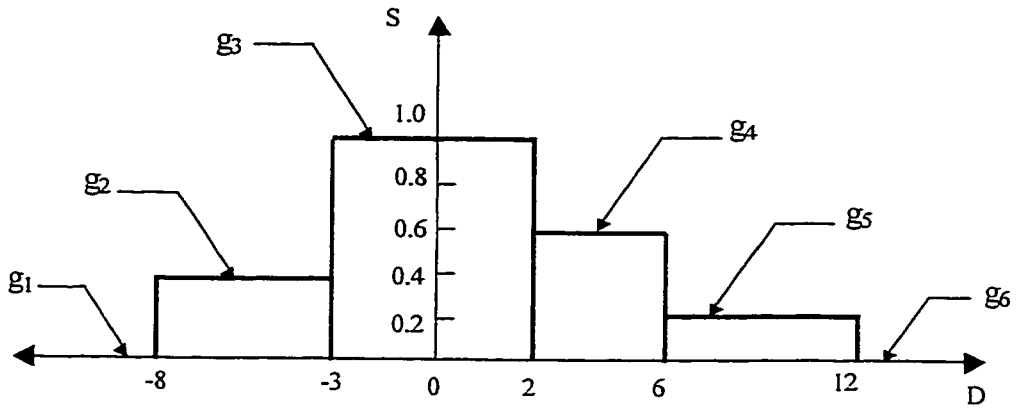


Figure 5.18: The step function representing similarity ranges

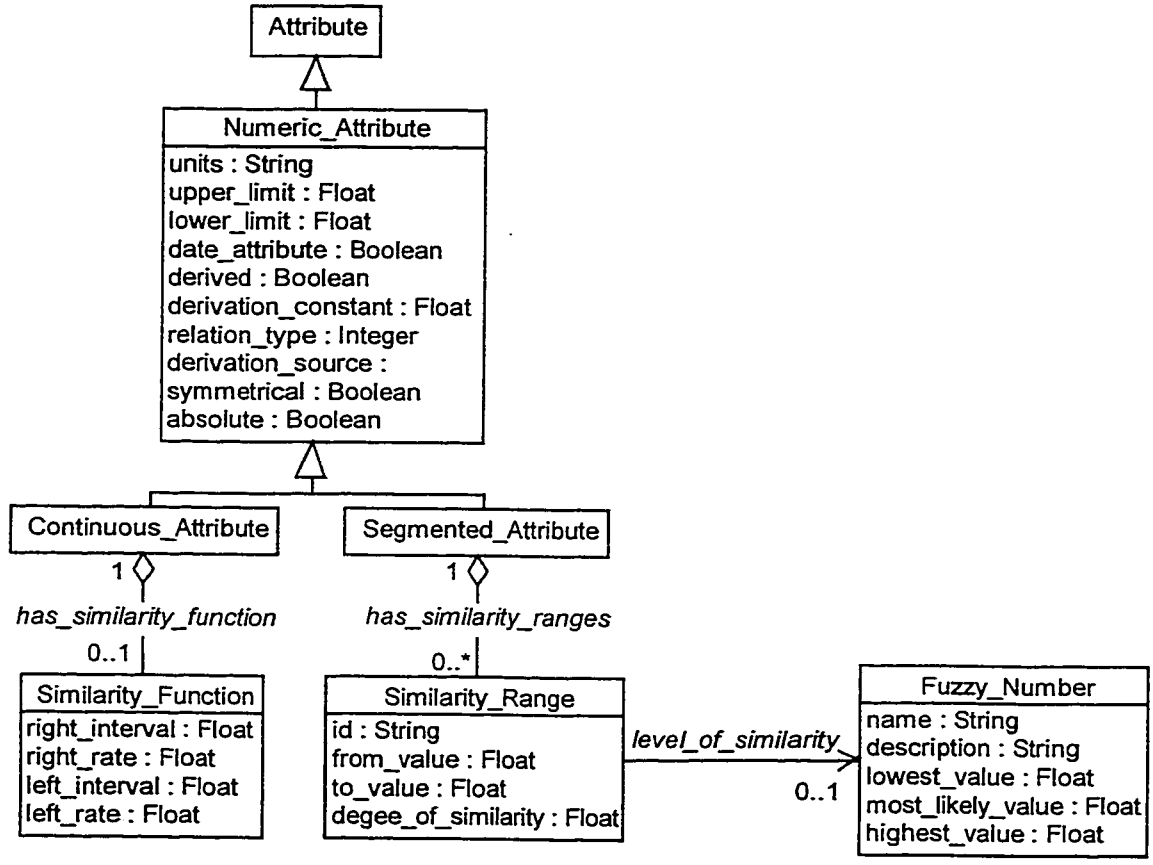


Figure 5.19: Representation of numeric attributes

5.4.1.2 Symbolic Attributes

Measuring the similarity between the values of a symbolic attribute is problematic. This is because the values of symbolic attributes cannot be represented on a unified scale like numeric values, and, consequently, the difference between these values cannot be calculated. Also, the values of a symbolic attribute can be represented in many formats or interpreted differently, which may cause problems in similarity calculations (Kolodner 1993). In order to avoid these problems, two types of symbolic attribute are defined: enumerated attribute and hierarchical attribute. These types are represented as classes inherited from the "Symbolic_Attribute" class: "Enumerated_Attribute" and "Hierarchical_Attribute". The definition of each attribute type along with the description of its similarity measure is provided below.

Enumerated Attributes

An enumerated attribute is a symbolic attribute that has a limited set of independent values. An example of such an attribute is the "Bridge Region" attribute. This attribute has the following four values for Quebec bridges: Northern (Nor), Eastern (Eas), Western (Wes), and Central (Cen). A similarity matrix is the most common way to define the similarity between the values of a symbolic attribute. This technique is similar to the similarity ranges for numeric attributes (Kolodner 1993). In similarity matrices, the degree of similarity between each pair of attribute values is defined. These values are represented as the row and the column of a matrix element that contains their degree of similarity. Assuming symmetrical similarity, degrees of similarity can be mirrored around the matrix diagonal that has a unit degree of similarity. Accordingly, the user only needs

to determine the degree of similarity for the upper or lower triangle of the similarity matrix. Table 5.2 shows an example of the similarity matrix of the "Bridge Region" attribute (only shaded cells are required). All matrix elements shown as crisp numbers can be replaced by fuzzy numbers to capture the uncertainty in evaluating the similarity.

	Nor	*Eas*	*Wes*	*Cen*
Nor	1.0	0.4	0.2	0.6
Eas	0.4	1.0	0.0	0.8
Wes	0.2	0.0	1.0	0.0
Cen	0.6	0.8	0.0	1.0

Table 5.2: The similarity matrix of the bridge region attribute

Hierarchical Attributes

A hierarchical attribute is a symbolic attribute whose values are represented as nodes of a taxonomy tree. A taxonomy tree (sometimes called abstraction hierarchy) is an n-ary tree in which the root is a hierarchical attribute and the nodes are all-possible values of that attribute (Kolodner 1993). Taxonomy tree technique differs from similarity matrix technique in terms of the knowledge it expresses. This technique expresses the relationship between attribute values through their position in the taxonomy tree (Bergmann 1998). By travelling towards the leaves of the tree, attribute values become more specialized. Figure 5.20 shows an example of a taxonomy tree for a symbolic attribute called "Bridge Element Material".

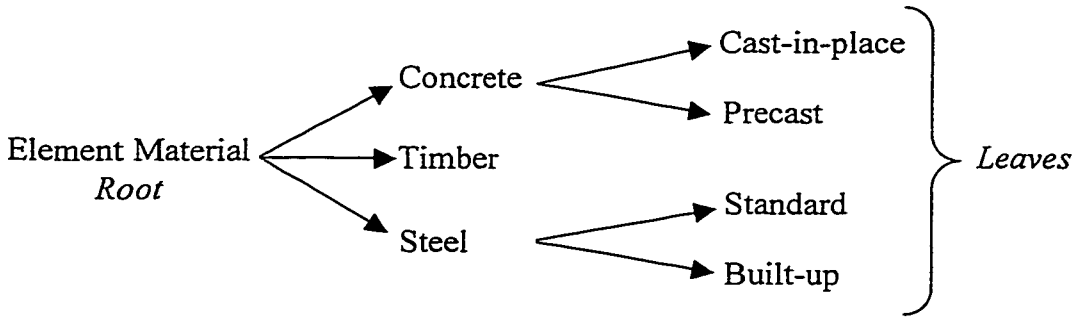


Figure 5.20: The taxonomy tree of a hierarchical symbolic attribute

The structure of the taxonomy tree implies that the similarity between any two nodes is directly proportional to the depth of their nearest common predecessor. For example, the similarity between "Cast-in-place Concrete" and "Precast Concrete" is greater than the similarity between "Precast Concrete" and "Standard Steel" because the nearest common predecessor of the former is "Concrete" while the nearest common predecessor of the latter is the Root. This kind of knowledge provided by the taxonomy tree about the similarity is not enough for calculating the degree of similarity between the different values. Therefore, numeric values should be incorporated with all the parent nodes to represent the degree of similarity among their child nodes. Adding these values in the previous example leads to Figure 5.21. In this figure the degree of similarity between "Cast-in-place Concrete" and "Precast Concrete" is estimated to be 0.4 while the degree of similarity between "Precast Concrete" and "Standard Steel" is estimated to be 0.

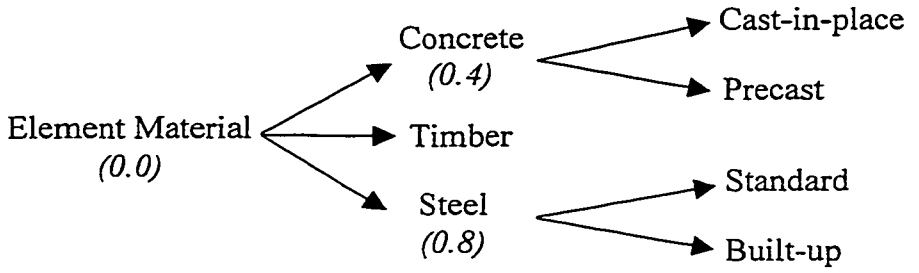


Figure 5.21: The taxonomy tree with the associated degrees of similarity

This method of estimating the degree of similarity between the taxonomy tree nodes is inaccurate because it expresses the similarity between all values having the same nearest common predecessor as one degree of similarity even though some of these values are closer than others. For example, assume both "Precast Concrete" and "Cast-in-place Concrete" are specialized into "Reinforced Concrete" and "Prestressed Concrete" as shown in Figure 5.22. The degree of similarity between "Reinforced Precast Concrete" and "Reinforced Cast-in-place Concrete" (0.4) is the same as the degree of similarity between "Reinforced Precast Concrete" and "Prestressed Cast-in-place Concrete", which is unrealistic. Also, representing the attribute value with a single node may cause confusion because attribute values may be abstracted in many different ways, which means that some nodes in the tree will be repeated under different parent nodes (Kolodner 1993). Therefore, some modifications need to be done to the taxonomy tree in order to increase the accuracy of the calculations and to determine what is the right branch of the tree to follow in case of repeated nodes.

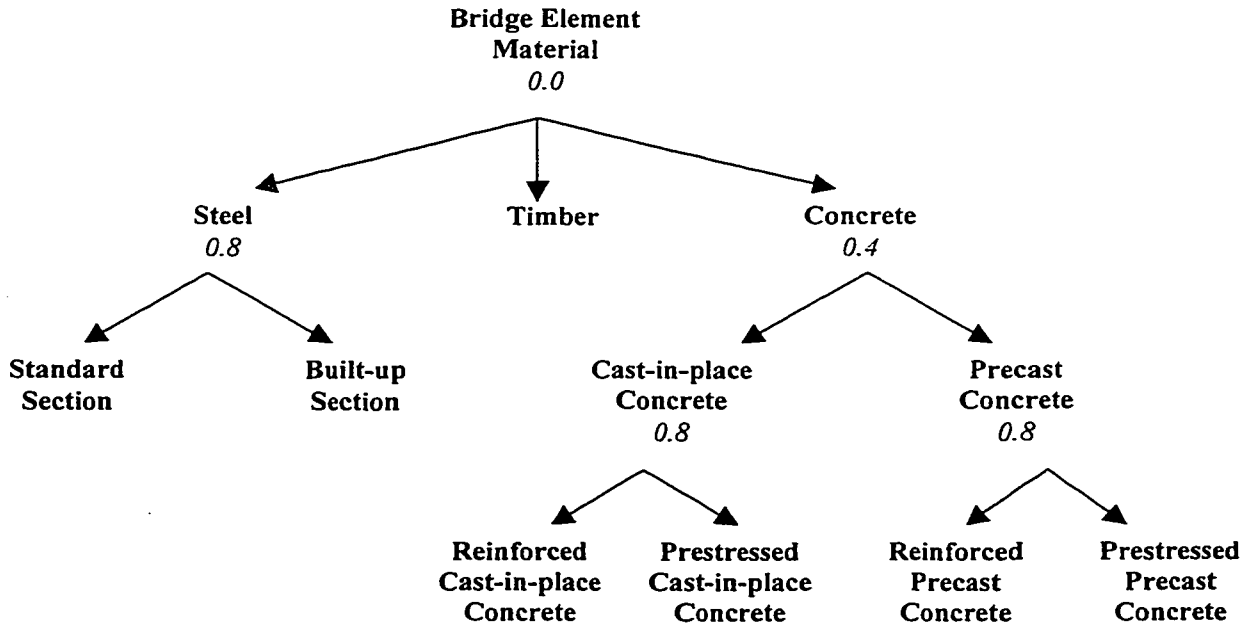


Figure 5.22: The conventional taxonomy tree of the bridge element material attribute

In the modified taxonomy tree, the decomposition of each parent node is based on a specific criterion. A numeric value is assigned to each criterion to represent the effect of this criterion on the degree of similarity among the child nodes. The child nodes resulting from the decomposition of different parents may be repeated if these parents are decomposed based on the same criterion. This organization of the tree nodes helps in identifying a list of criteria that should be used in comparing any two nodes. If the two nodes agree on a criterion, the similarity based on this criterion is equal to 1, otherwise the similarity should be the smaller of the numeric values associated with their parents. The net degree of similarity between the two values is equal to the multiplication of the degrees of similarity used for each criterion. In the modified taxonomy tree, all the numeric values associated with parent nodes can be replaced by fuzzy numbers.

Figure 5.23 shows the modified taxonomy tree for the same attribute used in Figure 5.22. In this tree, the bridge element material attribute has the same decomposition but its values are represented by the whole path starting from the root. That means a value such as cast-in-place reinforced concrete is represented as "Concrete>Cast-in-place> Reinforced" that results from following the path: "Concrete" to "Cast-in-place" to "Reinforced". This way of representation is more powerful than the one used in the conventional taxonomy tree because it allows having the same child node under different parents. It also makes the tree simpler even at high levels of decomposition. Using this methodology, the degree of similarity between "Concrete>Cast-in-place>Reinforced" and "Concrete>Precast>Reinforced" is calculated as 1 * 0.5 = 0.5 which represents the dissimilarity in the casting technique only. This degree of similarity is different from the

one between "Concrete>Cast-in-place>Reinforced" and "Concrete>Precast>Prestressed", which is calculated as 0.8 * 0.5 = 0.4. This value is more realistic because it considers the dissimilarity in casting the concrete and in reinforcing it. Figure 5.24 shows a part of the CBR shell design model that represents the retrieval knowledge of symbolic attributes.

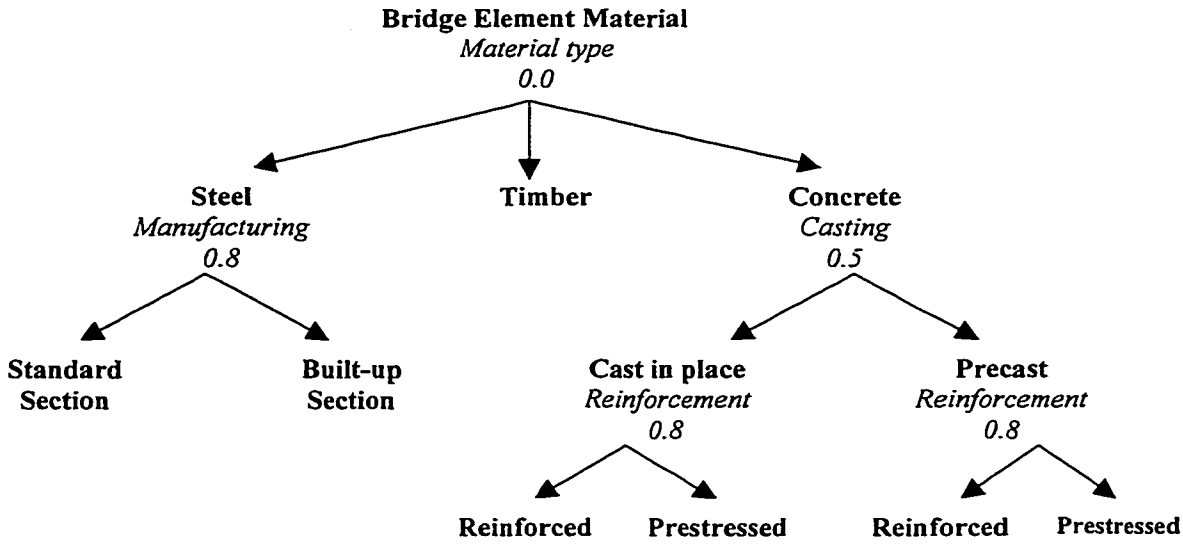


Figure 5.23: The modified taxonomy tree of the bridge element material attribute

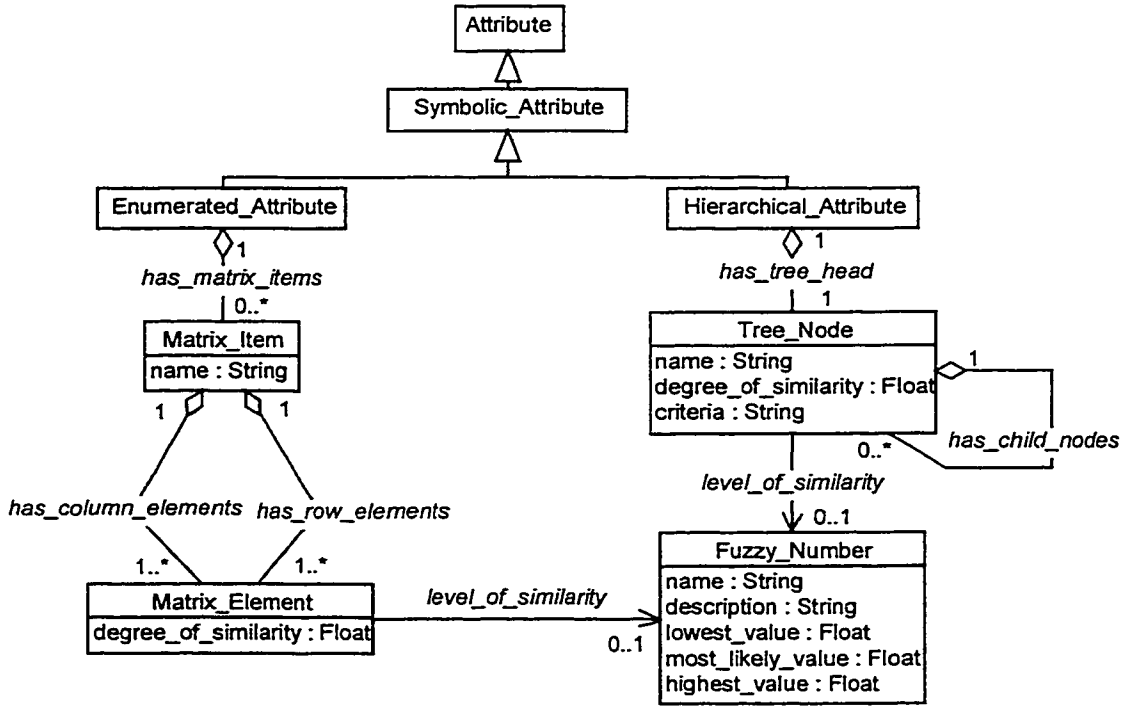


Figure 5.24: Representation of symbolic attributes

5.4.2 RETRIEVAL ALGORITHM

The role of the retrieval algorithm is to find the most similar cases to the query case. In the developed CBR shell, the retrieval algorithm starts when the user selects one domain entity to represent the query case. The algorithm continues by searching all domain entities that belong to the same domain entity type and matching them with the query case. Case matching is the process of comparing values of attributes that describe case contents. In the developed system, this process consists of four steps:

1. Matching static attributes of the domain entity
2. Matching time-dependent attributes of the domain entity
3. Matching attributes of child domain entities
4. Matching the attribute used for prediction

The first two steps in the matching process are called recursively in matching the child domain entities (i.e. step 3) because child domain entities include static and time-dependent attributes that describe their contents. While matching the attribute used for prediction (step 4) is not called recursively because each value of this attribute for a parent domain entity is the aggregation of the values of the same attribute for the child domain entities and, consequently, calling this matching recursively is redundant.

None of the above four matchings compares the attributes that belong to parent domain entities or parent time-dependent sets. Sharing attributes, sharing time-dependent sets, and deriving attributes provide ways to consider these attributes that belong to higher levels in the hierarchy.

In general, matching attributes follows different paths according to the properties of these attributes and the availability of their values. Non-discriminating attributes are disregarded completely in the matching process. Discriminating attributes may be indexing or non-indexing. For indexing attributes, whether they are exact-matching or not, the non-existence of a case value discards the whole case from the retrieval process, while the non-existence of a query value stops the whole retrieval process. This is because indexing attributes must have values in order to filter the retrieved cases. This filtering is carried out by skipping the case that includes at least one indexing attribute that does not have a full matching with the query case or at least one exact-matching attribute that does not have identical value to the query case. For non-indexing attributes, the non-existence of a case or query value results in similarities equal to 0 for these attributes. Table 5.3 shows the different paths followed by the retrieval algorithm according to the attribute role and the attribute value in both query and retrieved cases.

Attribute Condition					Retrieval Algorithm Path
No.	Discriminating	Indexing	Null in Query Case	Null in Retrieved Case	
1	Yes	Yes	No	No	Retrieve the case and calculate the similarity
2	Yes	Yes	Yes	No	Stop the retrieval process
3	Yes	Yes	No	Yes	Skip the case
4	Yes	Yes	Yes	Yes	Stop the retrieval process
5	Yes	No	No	No	Retrieve the case and calculate the similarity
6	Yes	No	Yes	No	Retrieve the case with attribute similarity = 0
7	Yes	No	No	Yes	Retrieve the case with attribute similarity = 0
8	Yes	No	Yes	Yes	Retrieve the case with attribute similarity = 0
9	No	No	No	No	Skip the attribute
10	No	No	Yes	No	Skip the attribute
11	No	No	No	Yes	Skip the attribute
12	No	No	Yes	Yes	Skip the attribute

Table 5.3: Different paths of the retrieval algorithm

Below are a detailed discussion of the four steps used in case matching and an explanation of how the similarity is calculated in each step.

5.4.2.1 Matching Static Attributes of the Domain Entity

Calculating the similarity among the attributes defined in the domain entities that represent cases is done using the following equations (Kolodner 1993):

$$AGS = TAS / TAW$$

$$TAW = \sum_{i=1}^{i=n} AW_i$$

$$TAS = \sum_{i=1}^{i=n} AS_i \times AW_i$$

Where,

AGS	Attribute global similarity
TAS	Total attribute similarity
TAW	Total attribute weight
AW_i	Weight of attribute i
AS_i	Local similarity between query and retrieved case values for attribute i calculated using the appropriate similarity measure

Example: Consider a domain entity type called "Bridge_Element" that has three attributes: "Material", "Span", and "Structural System". The properties of these attributes are listed in Table 5.4. Three bridge elements, "E01", "E02", and "E03" are defined to demonstrate the retrieval algorithm, assuming that the bridge element "E01" is the query

case and the bridge elements "E02" and "E03" are the stored cases. Table 5.5 shows attribute values for the three bridge elements.

#	Attribute Name	Attribute Type	Attribute Weight w_i	Attribute Role
1	Material	Symbolic – Hierarchical	0.6 Important	Indexing Discriminating
2	Span	Numeric – Segmented	0.2 Slightly Important	Not-Indexing Discriminating
3	Structural System	Symbolic – Enumerated	0.4 Fairly Important	Not-Indexing Discriminating

Table 5.4: Definition of attributes

Attribute Name	Query Case E01	Case 1 E02	Case 2 E03
Material	Prestressed Precast Concrete	Precast Concrete	Rolled Steel
Span	30	25	30
Structural System	Continuous	Continuous	Simple

Table 5.5: Definition of cases

Similarities among attribute values are calculated according to attribute types and the knowledge available about these types. The taxonomy tree presented earlier in Figure 5.22 is used to measure the similarity for the "Material" attribute, while the similarity ranges listed below in Table 5.6 are used to measure the similarity for the "Span" attribute. For the "Structural System" attribute, there are only two values, simple and continuous, that are completely dissimilar.

Span Segment ID	Range of Span Difference (%)	Degree of Similarity	Level of Similarity
S_1	$0 \leq D < 0.05$	1	Extremely Similar
S_2	$0.05 \leq D < 0.10$	0.6	Similar
S_3	$0.10 \leq D < 0.20$	0.2	Slightly similar
S_4	$0.20 \leq D$	0	Dissimilar

Table 5.6: Similarity ranges of the span attribute.

Using the above mentioned similarity measures, attribute values, and attribute properties, the similarity between "E01" and "E03" equals 0 because they are not matched in one of their indexing attributes (the "Material" attribute). The similarity between "E01" and "E02" is calculated as follows:

$$TAW = \sum_{i=1}^{i=3} AW_i = 0.6 + 0.2 + 0.4 = 1.20$$

$$TAS = \sum_{i=1}^{i=3} AS_i \times AW_i = 1 \times 0.6 + 0.2 \times 0.4 + 1 \times 0.2 = 0.88$$

$$AGS = TAS / TAW = 0.88 / 1.20 = 0.73$$

5.4.2.2 Matching Time-Dependent Attributes of the Domain Entity

The attribute global similarity *AGS* calculated above does not consider the time-dependent attributes used in describing case contents. Calculating the similarity among these attributes is problematic because such attributes take multiple values, each of which occurs at a different time point, constituting a time series. The time series of the query case often differs from its corresponding in the retrieved case with respect to number of points and corresponding time values. In order to compare such attributes, the algorithm should compute the time of the retrieved case at which the prediction will be carried out.

This time is called the prediction time and is determined as shown in Figure 5.25. It is the time at which the value of the attribute used for prediction in the retrieved case is the same as the value of that attribute in the query case at the current time. Linear relationships are assumed between the consecutive values of that attribute.

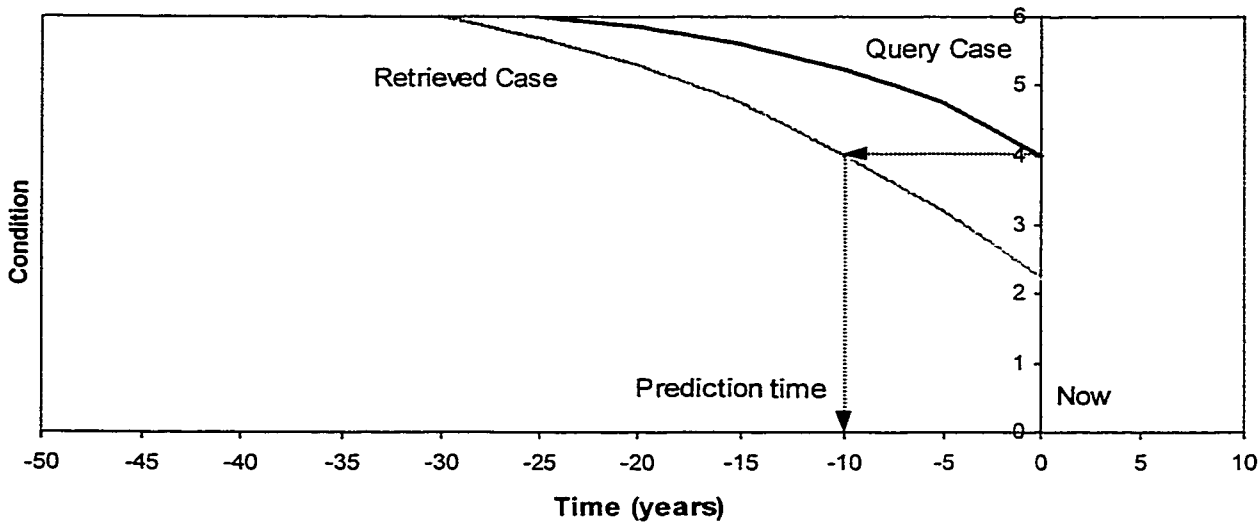


Figure 5.25: Determination of the prediction time

After obtaining the prediction time, the similarity between the value of a time-dependent attribute in the retrieved case at the prediction time and the value of the same attribute in the query case at the current time can be calculated by two methods. The first method is used if the attribute has numeric values that can be obtained at any time by interpolation and extrapolation assuming linear relationship between the consecutive attribute values. Examples of such attributes are the bridge traffic and the material condition of the interacting entities. The second method is used if the attribute has symbolic values given at specific time points and cannot be interpolated, such as the maintenance actions. In this method, the attribute value that is the closest to the prediction time in the retrieved case and the attribute value that is closest to the current

time in the query are obtained. The similarity between these two values is calculated, using an appropriate similarity measure, and multiplied by the similarity between their corresponding time points to compensate for the difference in their timings. Weights and similarities of all time-dependent attributes (except the one used for prediction) are added to *TAW* and *TAS* to be considered in the *AGS* that is calculated before. It should be noted that matching time-dependent attributes does not consider matching their past values because the effect of these values is already captured by the past values of the attribute used for prediction.

Example: Consider that two time-dependent set types, "Bridge_Traffic" and "Element_Condition", are added to the previous example. The "Bridge_Traffic" has two attributes: "Measuring Date" (time attribute) and "ADT" (time-dependent attribute), while the "Element_Condition" has two attributes: "Inspection Date" (time attribute) and material condition rating "MCR" (attribute used for prediction). The properties of these attributes are listed in Table 5.7. Time-dependent sets defined for bridge elements "E01" and "E02" are also listed in Table 5.8 along with the values of their attributes.

Attribute Name	Time-Dependent Set Type	Attribute Type	Attribute Weight *AW*	Attribute Role
Measuring Date	Bridge Traffic	Numeric – Continuous		Time Attribute
ADT	Bridge Traffic	Numeric – Continuous	0.2 Slightly Important	Not-Indexing Discriminate
Inspection Date	Element Condition	Numeric – Continuous		Time Attribute
MCR	Element Condition	Numeric – Continuous	1.0 Extremely Important	Used for Prediction

Table 5.7: Definition of time-dependent attributes.

Attribute Name	Query Case E01	Case 1 E02
Measuring Date	(1990, 1995)	(1988, 1992, 1996)
ADT	(2000, 2500)	(1500, 2000, 2400)
Inspection Date	(1990, 1993, 1996)	(1987, 1991, 1994, 1996)
MCR	(6,5,5)	(6,5,5,4)

Table 5.8: Definition of time-dependent sets and their values.

To consider time-dependent attributes in calculating *AGS*, the prediction time has to be determined. According to Figure 5.25, the prediction time for case "E02" is 1994 (the time at which the value of "MCR" in case 1 equals the current value of "MCR" in the query case, which equals 5). At that time, the values of the "ADT" attribute in "E01" and "E02", obtained by interpolation, are: 2600 and 2200 respectively. Assuming that the similarity function of the "ADT" attribute is symmetrical and has a relative interval equals 0.2 and a rate equals 1, the attribute degree of similarity will be:

$$\left(\frac{2600 * 0.2 - |2200 - 2600|}{2600 * 0.2}\right)^1 = 0.23$$

5.4.2.3 Matching Attributes of Child Domain Entities

The similarity calculated above assumes that the domain entity is not decomposed into smaller domain entities. In complex domains, such as bridge management, many domain entities are broken down into smaller domain entities to resolve the complexity of the domain (refer to the hierarchical decomposition of bridges in section 4.2). In this situation, similarity calculations between the query case domain entity and retrieved case domain entities should be adjusted to account for their child entities (sub-cases).

Computing the similarity between hierarchically decomposed cases is carried out recursively in a bottom-up approach (Bergmann and Stahl 1998). First, similarities between the attributes of child entities are calculated and aggregated using attribute weights to determine attribute global similarities between these child entities (sub-cases) Then, these attribute global similarities are aggregated using child entity weights (balance factors) to determine what is called sub-case global similarity *SubGS*. The similarity between parent entities, considering their decomposed entities, is called decomposition global similarity *DGS* and is defined as shown in the following equations:

$$DGS = TDS / TDW$$

$$TDW = 1 + W_{agg}$$

$$TDS = AGS + SubGS * W_{agg}$$

Where,

DGS	Decomposition global similarity between parent entities
TDW	Total decomposition weight
TDS	Total decomposition similarity
SubGS	Sub-case global similarity
W_{agg}	Aggregation weight of the case domain entity type.

In the previous equations, matching child domain entities *SubGS* is added to matching attributes of parent entities *AGS* to provide the decomposition global similarity *DGS*. This addition is carried out using the aggregation weight of the case domain entity type that controls the relative importance between the two matchings assuming that the importance of matching case attributes is equal to 1.

Calculating the sub-case global similarity between a query parent entity and a retrieved parent entity is problematic (Flemming 1994). This is because, different number of decomposition levels, different number of domain entity types at each decomposition level, and different number of domain entities from each domain entity type may exist in the query and retrieved cases as shown in Figure 5.26. Moreover, there is no clear rule that determines which of the retrieved child entities should be matched against every query child entity. In order to solve these problems, a decomposition similarity matrix is formulated for every domain entity type in every decomposition level. In this matrix, similarities between query case child entities, represented by matrix rows, and retrieved case child entities, represented by matrix columns, are calculated. Table 5.9 shows the decomposition similarity matrix for the general example shown in Figure 5.27. The two child entities that have maximum similarity between them, in this matrix, are matched and removed from the matrix. This step is repeated till child entities of either the query case or the retrieved case, whose number is fewer, are all removed. Then, the weighted sum of the obtained similarities is computed and divided by the sum of weights of all child entities of either the query or the retrieved case, whichever is greater, in order to account for the unmatched child entities as shown in the following equations:

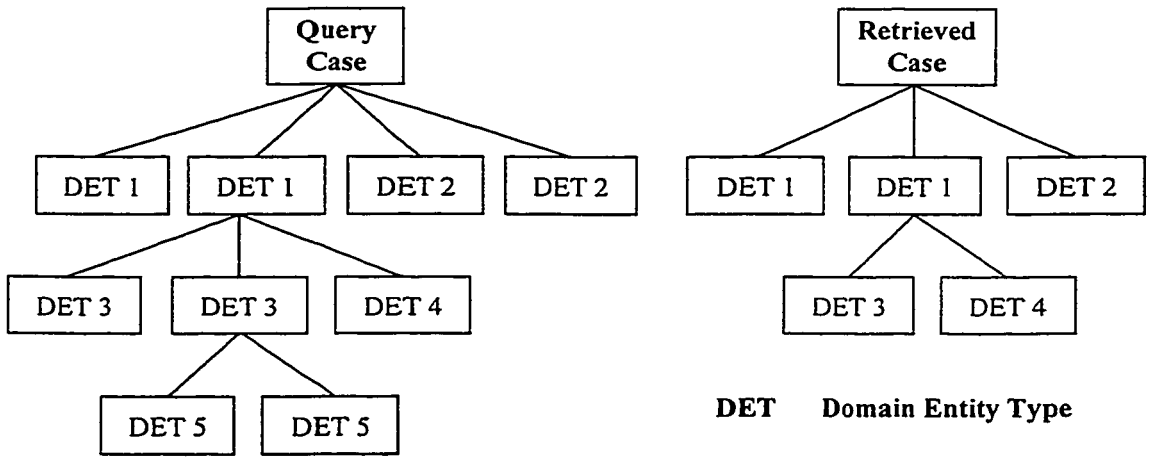


Table 5.26: The incompatibility between the structure of query and retrieved cases

	$S^{r_i}{}_1$	$S^{r_i}{}_2$	$S^{r_i}{}_3$	$S^{r_i}{}_4$
$S^q{}_1$	$DGS(S^q{}_1, S^{r_i}{}_1)$	$DGS(S^q{}_1, S^{r_i}{}_2)$	$DGS(S^q{}_1, S^{r_i}{}_3)$	$DGS(S^q{}_1, S^{r_i}{}_4)$
$S^q{}_2$	$DGS(S^q{}_2, S^{r_i}{}_1)$	$DGS(S^q{}_2, S^{r_i}{}_2)$	$DGS(S^q{}_2, S^{r_i}{}_3)$	$DGS(S^q{}_2, S^{r_i}{}_4)$
$S^q{}_3$	$DGS(S^q{}_3, S^{r_i}{}_1)$	$DGS(S^q{}_3, S^{r_i}{}_2)$	$DGS(S^q{}_3, S^{r_i}{}_3)$	$DGS(S^q{}_3, S^{r_i}{}_4)$

Table 5.9: Decomposition similarity matrix.

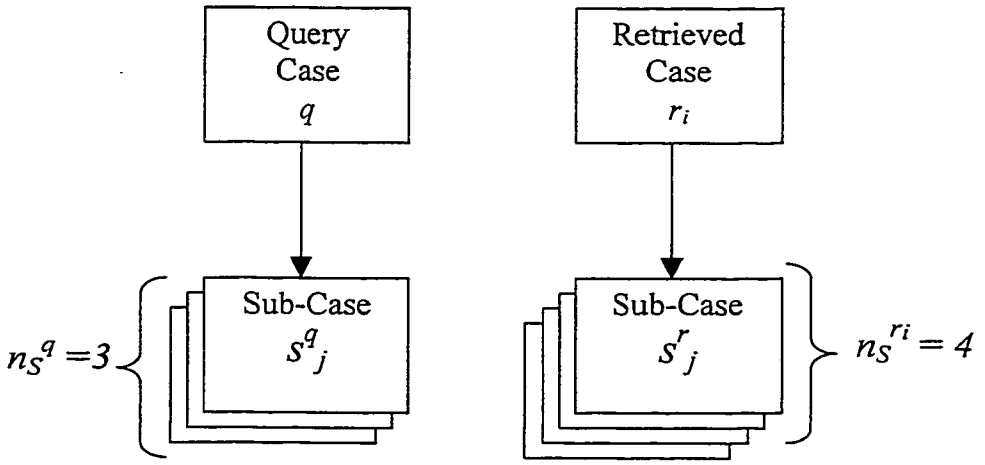


Figure 5.27: General example of decomposed cases

$$SubGS = TSubS \; / \; TSubW$$

$$TSubS = \sum_{j=1}^{j=n_S^y} W_{ent}^{S^y_j} * DGS\ (S^x, S^y)_{\max}\ .$$

$$TSubW = \sum_{k=1}^{k=n_S^x} W_{ent}^{S^x_k}$$

Where,

$TSubS$	Total sub-cases similarity
$TSubW$	Total sub-cases weights
$n_S{}^q$	Number of sub-cases of the parent case q

$n_S{}^r{}_i$	Number of sub-cases of the parent case r_i
x	Parent case (q or r_i) that has the greater number of sub-cases
y	Parent case (q or r_i) that has the smaller number of sub-cases
S^x, S^y	Sub-cases of the cases x and y respectively that provide maximum *DGS* and have not been used in matching other sub-cases
W^S_{ent}	Entity weight of a sub-case S

Example: Consider the two foundation assemblies shown in Figure 5.28. The query assembly consists of two piles and one pile cap, while the retrieved assembly consists of three piles and one pile cap. Table 5.10 shows the decomposition similarity matrix of these assemblies along with the entity weights of the sub-cases.

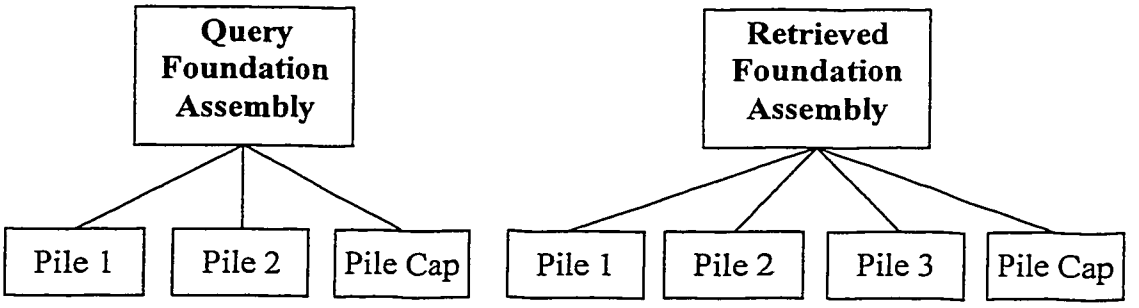


Figure 5.28: The hierarchical decomposition of two foundation assemblies

Sub-Cases (W_{ent})	***Pile1 (0.6)***	***Pile2 (0.6)***	***Pile3 (0.6)***	***Pile Cap (0.4)***
Pile1 (0.6)	*0.30*	*0.6*	*0.45*	*0*
Pile2 (0.6)	*0.8*	*0.25*	*0.5*	*0*
Pile Cap (0.4)	*0*	*0*	*0*	*0.75*

Table 5.10: The decomposition similarity matrix of two foundation assemblies

$$SubGS = (0.6 \times 0.6 + 0.8 * 0.6 + 0.75 * 0.4) / (0.6 + 0.6 + 0.6 + 0.4) = 0.52$$

5.4.2.4 Matching the Attribute Used for Prediction

Comparing values of the attribute used for prediction in both query and retrieved cases is the last step in the case matching process. This attribute requires special consideration because its contribution in determining the overall case similarity varies according to the availability of its records. It is also the only time-dependent attribute that participates in the case matching process with more than one value (time series) each of which has a different importance according to its time order.

Therefore, some properties have been defined especially for such an attribute in order to facilitate its matching calculations. One of these properties is the maximum multiplicity, which determines the number of time points that are sufficient for comparing time series. The domain expert has to define the number that satisfies the system requirements depending on his own judgement and experience. Another property is the weight of the attribute used for prediction, which is different from the weights of static attributes because of its dynamic nature. This weight changes its value according to the number of points used for the time series comparison. It reaches its maximum value when the number of points equals the maximum multiplicity. The domain expert has to define this maximum value according to the relative importance of the attribute used for prediction with respect to the other attributes. Another property is the time-weight factor that controls relative importance among the different points of a time series. This factor takes a value between 0 and 1 and determines how the attribute weight declines with time. This declination is assumed to be exponential, as shown in Figure 5.29, based on the importance of the condition history in predicting future conditions discussed in

et al. (1997) and Madanat and Ibrahim (1995). This assumption deserves further investigation in future research in order to be accepted. The equation that calculates the weight of each time-dependent set that contains the attribute used for prediction according to its time order is:

$$SetW = SetW_{\max} \times \left(\left(\frac{SetN}{SetN_{\max}} \right)^{SetF} - \left(\frac{SetN - 1}{SetN_{\max}} \right)^{SetF} \right)$$

Where,

$SetW$	Weight of the time-dependent set of the attribute used for prediction
$SetW_{max}$	Maximum set weight
$SetN$	Set timely order (1 for the newest set)
$SetN_{max}$	Maximum multiplicity
$SetF$	Time-weight factor

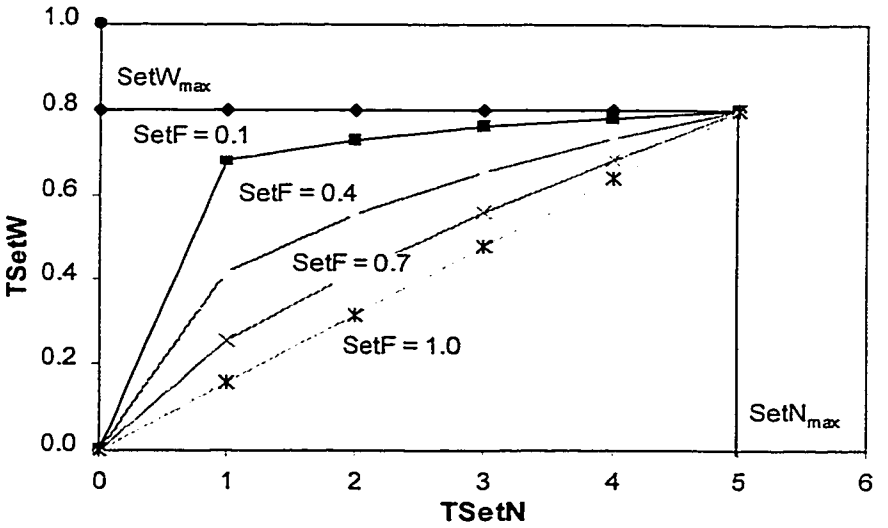


Figure 5.29: Changing total set weight with the total number of sets at different time-weight factors

Calculated weights of matched sets are aggregated to provide the total set weight *TSetW*, as shown in Figure 5.29, which is equal to or less than the maximum weight depending on the availability of points and the time weight factor. In order to come up with the case global similarity *CGS*, similarities between corresponding points in the query time series and the case time series are calculated, aggregated, and added to the decomposition global similarity as follows:

$$
\begin{aligned}
CGS &= TCS \;/\; TCW \\
TCS &= DGS + SetGS \times TSetW \\
SetGS &= TSetS \;/\; TSetW \\
TSetS &= \sum SetS \times SetW \\
TSetW &= \sum SetW \\
TCW &= 1 + TSetW
\end{aligned}
$$

Where,

CGS	Case global similarity
TCS	Total case similarity
TCW	Total case weight
SetGS	Set global similarity
TSetS	Total set similarity
TSetW	Total set weight
SetS	Similarity of the time-dependent set of the attribute used for prediction

Example: Use the example that is previously mentioned in sub-section 5.4.2.2 (Matching the time-dependent attributes of the domain entity). Assume that the maximum

set weight is 1.0, maximum multiplicity of the "MCR" attribute is 4 and the time-weight factor is 0.4. Table 5.11 shows the calculated weight and similarity for every set along with the total set weight and the total set similarity.

SetN	Query Value	Case Value	SetS	SetW	SetW x SetS
1	5	5	1	0.57	0.57
2	5	5	1	0.18	0.18
3	6	5.75	0.75	0.13	0.10
			TOTAL	*0.88*	*0.85*

Table 5.11: Calculating the similarity of the attribute used for prediction

In order to calculate the case global similarity, the similarity of time-dependent attributes calculated in section 5.4.2.2 is added to the *AGS* calculated in section 5.4.2.1 as follows:

$$AGS = (0.23 + 0.88) / (0.2 + 1.2) = 0.79$$

For simplicity purposes, matching child domain entities is ignored in this example. This means that the decomposition global similarity *DGS* is equal to the *AGS* calculated above. The case global similarity *CGS* can be obtained by adding the similarity of the attribute used for prediction presented in Table 5.11 as follows:

$$SetGS = TSetS / TSetW = 0.85 / 0.88 = 0.97$$

$$TCS = DGS + SetGS * TSetW = 0.79 + 0.97 \times 0.88 = 1.54$$

$$TCW = 1 + TSetW = 1 + 0.88 = 1.88$$

$$CGS = TCS / TCW = 1.54 / 1.88 = \mathbf{0.82}$$

5.4.2.5 Retrieval Parameters

Some parameters control the retrieval algorithm and its results. These parameters are: retrieval threshold, which eliminates retrieved cases that have a case global similarity less than a specific value; top retrievals, which determines the maximum number of best matched cases to be retrieved; maximum decomposition levels, which determines how deep the retrieval algorithm can go in the case hierarchical decomposition; and prediction period, which specifies the time at which the predicted value will be obtained. These parameters in addition to some other administrative information such as retrieval operation id, description, and date are represented as data members of a class called "Retrieval_Operation".

The "Domain_Entity" class has a one-to-many bi-directional aggregation relationship with the "Retrieval_Operation" class to define the retrieval operations that have been carried out for every domain entity. This relationship is called "has_retrieval_operations". Every retrieval operation also stores the current time and the current value of the attribute used for prediction in the query case. It also stores the average and the standard deviation of predicted values and the most frequent predicted value in the retrieved cases. The "Retrieval_Operation" class has a one-to-many bi-directional aggregation relationship with a class called "Retrieved_Case". This relationship is called "has_retrieved_cases". Every object instantiated from the "Retrieved_Case" class represents one retrieved case resulting from the retrieval operation. It stores the predicted value, the prediction time, the case global similarity, and all intermediate calculations as shown in Figure 5.30. Every retrieved case must point to the domain entity that has been retrieved. It can also

store the results of the adaptation process in its data member "adaptation_results", which will be discussed in the following section.

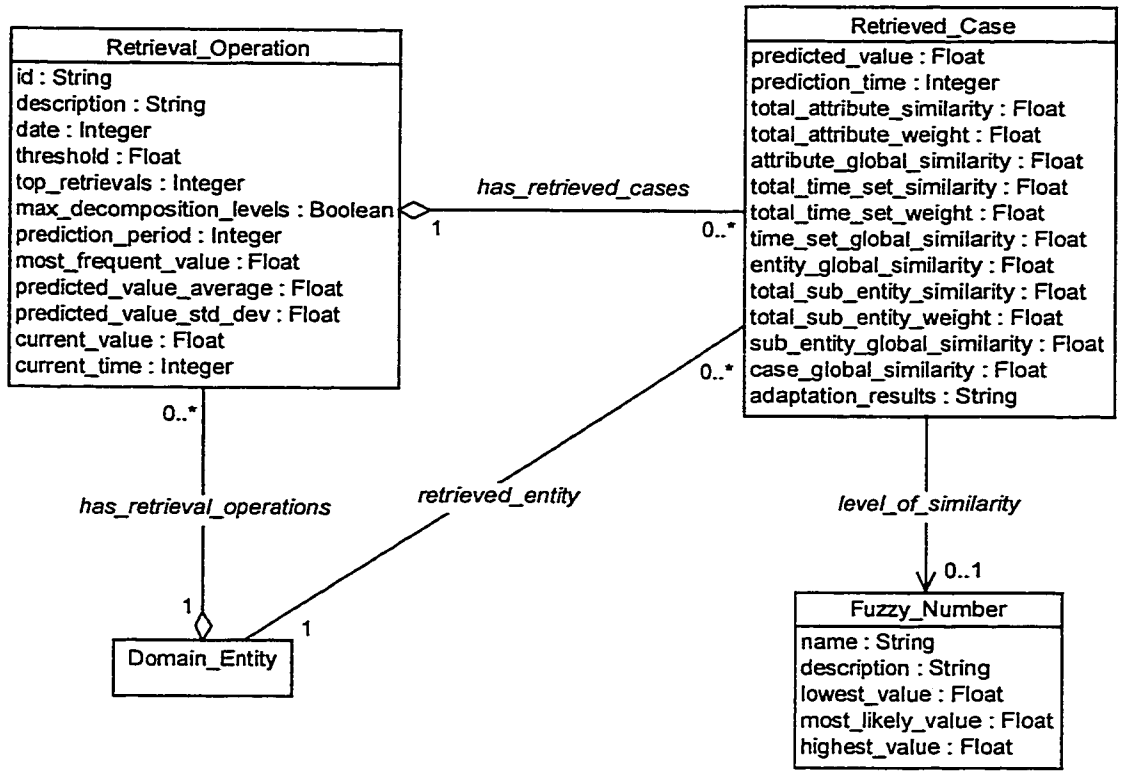


Figure 5.30: Representation of retrieval operations and retrieved cases.

5.5 CASE ADAPTATION

The cases returned from the retrieval process rarely match the query case exactly. Slight differences often exist between the description of the query case and the description of the retrieved cases including the case that has the best match (Roddis and Bocox 1997). Therefore, some changes have to be made to the solution of the retrieved cases in order to fit the query case. These changes can be made automatically by the system or manually by the user. The process of carrying out these changes is known as case adaptation (also as case modification) (Maher et al. 1995).

Automatic adaptation of the retrieved cases is problematic. Recognizing the differences between the query case and the retrieved case, how these differences affect the retrieved case solution, and how the retrieved solution can be corrected for these differences, is not a simple task (Kolodner 1993). Therefore, many CBR systems do not consider the automatic adaptation of the retrieval results. Examples of these systems are: the CBR system for selecting retrofitting methods of fatigue damage on steel bridges (Tanaka et al. 1996), the CBR system for resolving fabrication errors in steel highway bridges (CB-BFX) (Roddis and Bocox 1997), and the case-based design system in the SEED (Software Environment to support the Early phases of building Design) system (Flemming 1994). Such systems retrieve similar cases only and let the user adapt the solution to fit the current problem using his/her engineering knowledge (manual adaptation). However, some of these systems provide the user with additional information that helps in adapting the solution successfully as in Tanaka et al. (1996). Other systems provide the user with efficient tools that allows him/her to navigate through the case components and modify them interactively and dynamically as in Flemming (1994).

There are four major categories of the automatic adaptation methodologies: by substitution, by transformation, by derivational replay, and special-purpose adaptation (Kolodner 1993). Adaptation by substitution is the process of replacing the values that belong to the query case in the retrieved case (Maher et al. 1995). This method is appropriate if the attributes for which the values will be substituted exist in both the query and the retrieved cases. But, if these attributes do not exist, adaptation by

transformation should be used to delete, insert, or adjust these attributes in the retrieved case (Kolodner 1993). An example of implementing adaptation by substitution is the CBR shell called CBR-Works 4. In this shell, there are two kinds of rules: completion rules, and adaptation rules. These rules have been employed to integrate the general knowledge of the domain with the reasoning process (Bergmann el al. 1996). An example for implementing adaptation by transformation is the CBR shell called CASETOOL. In this shell, both problem-specific knowledge and problem-independent knowledge are utilized in the form of rules (Kumar and Krishnamorthy 1995). Derivational replay is the adaptation methodology that re-computes the new solution by following the same process that was used in computing the old solution. This methodology is powerful in solving design problems because it records the complex design decisions generated throughout the design process (Rivard et al. 1998). However, this methodology is challenging because it cannot be applied unless the old cases contain complete knowledge about how the past solutions are derived as well as the intermediate results (Kumar and Krishnamorthy 1995). For an example of implementing adaptation by derivational replay, see the CBR system called SEED-Config (Rivard et al. 1998).

The adaptation method is selected based on the problem domain. In some domains, the above-mentioned methodologies are considered as inefficient (Kolodner 1993). Many CBR systems have introduced special-purpose adaptation methods, each of which can deal with a specific problem efficiently. An example of the special-purpose adaptation is the design case adaptation using the constraint satisfaction formulation. This formulation

was used in the CBR system called CADSYN. This adaptation uses heuristic search to eliminate constraint violations in the potential solution (Maher et al. 1995).

Even though the primary source of knowledge in the CBR system is the cases stored in the case library, additional knowledge is required to repair the solutions of the partially matched cases. The need for this type of repair depends on three factors:

1. **The task of the CBR system.** In synthesis tasks such as design and planning, adaptation is much more needed than for decision support tasks such as classification, diagnosis, and prediction (Bergmann 1998).
2. **The size of the system case library.** The more the cases in the case library, the less the possibility of having a novel case that requires too much adaptation (Leake 1996).
3. **The limits used to control the retrieval process.** The need for adaptation can be decreased by narrowing the upper and lower limits of the indexing attributes (Kumar and Krishnamorthy 1995).

The choice of the appropriate adaptation methodology depends mainly on the nature of the retrieved cases and the availability of the domain knowledge. Derivational replay adaptation is recommended for the cases that have composite and knowledge-intensive solutions, such as design cases. This methodology cannot be used unless there is plenty of domain knowledge available from domain experts. Adaptation by transformation is recommended for the cases that have less complicated solutions and require less domain

knowledge. Adaptation by substitution is recommended for the cases that have simple solutions and whenever the domain knowledge is rare.

Since the purpose of the developed system is prediction, the case library contains thousands of cases, and matching limits can be customized to control the retrieval process, only simple adaptation capabilities are required. The adaptation by substitution methodology has been selected for two reasons. First, the solution of retrieved cases, which is represented as condition ratings of infrastructure facility components, is simple. Second, domain knowledge regarding the deterioration of infrastructure facilities is rare. This is due to the multiplicity of deterioration factors that are sometimes inter-related and unobserved, such as the design and construction quality. The adaptation by substitution will be applied by means of simple adaptation rules that are obtained from the domain literature or experts. The following sections discuss how the adaptation rules are stored in the developed system, how they are processed, how they are maintained, and how they are presented to the user.

5.5.1 REPRESENTATION OF ADAPTATION KNOWLEDGE

Adaptation rules are used to modify the solution of a retrieved case to fit the current problem. Each adaptation rule is represented as an object instantiated from a class called "Adaptation_Rule", and stored in the database sub-module called "Adaptation Knowledge Base". A one-to-many bi-directional aggregation relationship called "has_adaptation_rules" has been introduced between the "Domain_Entity_Type" class and the "Adaptation_Rule" class as shown in Figure 5.31 to determine the adaptation

rules defined for every domain entity type. Each adaptation rule has a unique identifier stored in a data member called "id". It also has a data member called "description" that stores rule name, description, and function.

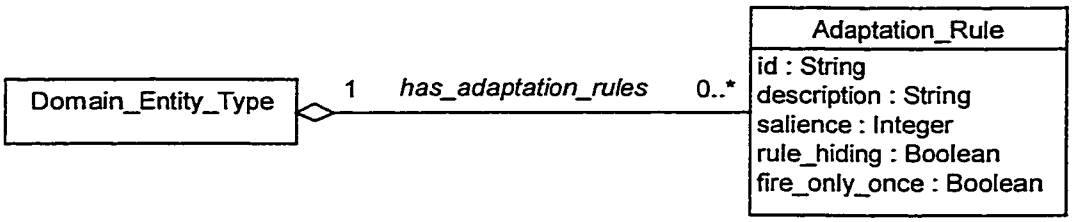


Figure 5.31: Representation of adaptation rules

Adaptation rules are of the form IF-THEN. When the left-hand side (the IF part) of many rules is satisfied, these rules become active. The sequence of firing active rules can be determined by means of rule salience: the rule with greater salience has a priority in firing over the rule with smaller salience. The data member called "salience" is used to store the salience value. Also, Boolean data members called "rule_hiding" and "fire_only_once" are used, respectively, to remove a rule temporarily from the knowledge base and to restrict the number of rule firing to one.

An adaptation rule has a set of conditions (the IF part) that describe the situation in which this rule is applicable and another set of actions (the THEN part) that describe modifications resulting from applying this rule. Rule conditions and actions are represented as objects instantiated, respectively, from the two classes called "Rule_Condition" and "Rule_Action" shown in Figure 5.32. These two classes are inherited from a class called "Rule_Expression" that contains the common features of rule conditions and actions. Due to the multiplicity of conditions and actions of an

adaptation rule, each rule expression has a unique identifier stored in a data member called "id". It also has a data member called "type" to determine whether this expression is a rule condition or a rule action.

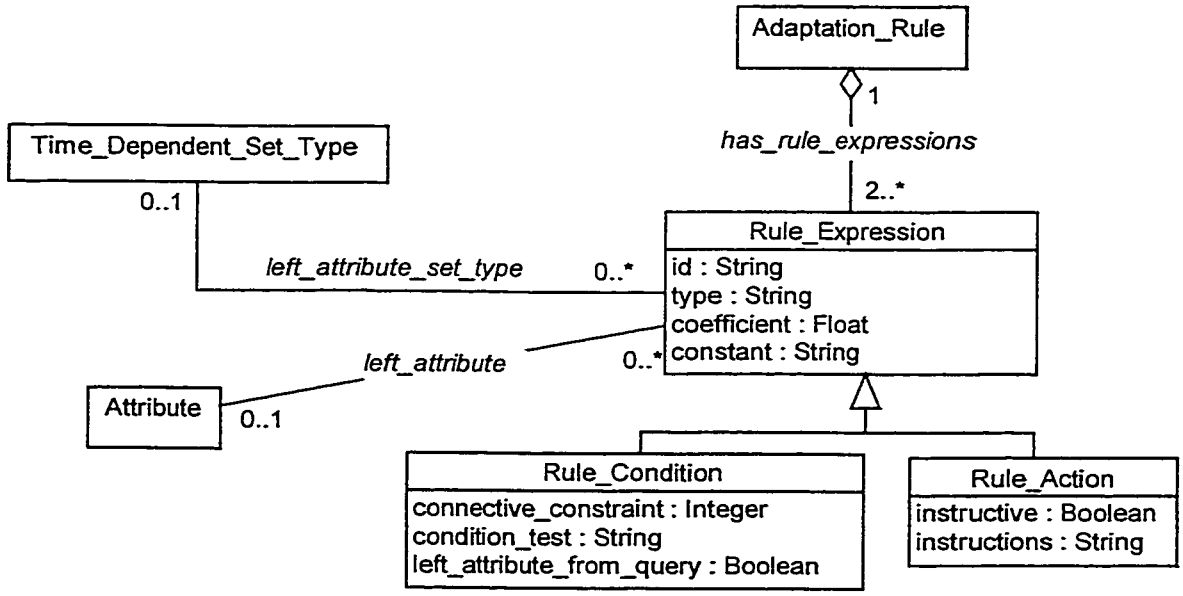


Figure 5.32: Representation of rule expressions

Whether a rule expression is a condition or an action, it is represented as a simple linear expression that acts a test formulation in the case of rule conditions and an assignment formulation in the case of rule actions. The left argument of this linear expression is the value of one of the attributes defined for the domain entity type that contains the adaptation rule. In situations where values from more than one attribute are required to construct the rule expression (i.e. polynomial expression), derived attributes can be defined, first, to aggregate the values of many attributes and, then used in the simple linear expression.

The test formulation that checks the satisfaction of a rule condition is represented as follows:

$$Y\ (test)\ a\ X + b$$

Where,

Y	Value of a specific attribute in the query or retrieved case
(test)	Condition test such as =, ≠, <, >, ≤, or ≥
X	Value of the same attribute in the opposite case
a	Expression coefficient
b	Expression constant

Below is an example for a test formulation in an adaptation rule defined for the "Deck" domain entity type. This formulation checks that the difference between the values of the "Span_Length" attribute in the retrieved case and the query case is greater than or equal to 20% of the attribute value in the query case.

$$[\text{Span\_Length}]_{\text{retrieved case}} >= 1.2 * [\text{Span\_Length}]_{\text{query case}} + 0$$

Data members called "left_attribute_from_query" and "condition_test" have been introduced into the "Rule_Condition" class to determine, respectively, the source of the attribute value used as a left argument and the condition test. Also, a data member called "connective_constraint" has been introduced into the same class to determine the type of connection (e.g. And, or Or) among different rule conditions.

The assignment formulation that modifies the value of an attribute in the retrieved case is represented as follows:

$$Y = aX + b$$

Where,

Y	Value of a specific attribute in the retrieved case
X	Value of the same attribute in the query case or the old value of the same attribute in the retrieved case when the attribute is used for prediction
a	Expression coefficient
b	Expression constant

Below is an example for an assignment formulation in an adaptation rule defined for the "Deck" domain entity type. This formulation modifies the value of the "Material_Condition" attribute in the retrieved case by subtracting 0.5 from its old value.

$$[\text{Material\_Condition}]_{\text{retrieved case}} = 1 * [\text{Material\_Condition}]_{\text{retrieved case}} - 0.5$$

In both test and assignment formulations, expression coefficient can be used only with numeric attributes while expression constant can be used with both symbolic and numeric attributes. The expression coefficient is stored in a float data member called "coefficient" and the expression constant is stored in a string data member called "constant". Attributes used in the left argument of any adaptation rules that are defined for a domain entity type must be selected from either that entity type or any of its time-dependent set types. The selected time-dependent set type is determined through a many-to-one bi-directional association relationship between the "Rule_Expression" class and the

"Time_Dependent_Set_Type" class. This type of relationship is called "left_attribute_set_type" and is used only whenever the left argument is selected from a time-dependent set type.

Rule actions may have two results: instructing the system user, or assigning a value to one of the retrieved case attributes. A Boolean data member called "instructive_action" has been introduced in the "Rule_Action" class to distinguish instructive actions from assigning ones. A collection of both action types may constitute the right-hand side of any adaptation rule. For instructive actions, the data member called "instructions" is used to store what the system is advising the user.

Sometimes, complex adaptation rules that cannot be represented using the object-oriented design model shown in Figure 5.32 are needed. In these situations, additional rules are created by the domain expert using the syntax of the expert system programming language CLIPS. These rules are not stored in the database, but they are stored in a separate text file and appended to the rules stored in the database during the adaptation process.

5.5.2 ADAPTATION PROCESS

After adaptation rules are defined for every domain entity type, the case-based reasoner is ready to transform these rules into CLIPS constructs. The C Language Integrated Production System (CLIPS) is an expert system programming language that has been developed by the Artificial Intelligence Section of NASA's Johnson Space

Center (CLIPS 6.10 1998b). Constructs are one of the basic programming elements in CLIPS, which are considered as the building blocks of the system knowledge base. There are different types of constructs that can be defined in CLIPS such as "deftemplate", "defrule", "deffacts", "deffunction", "defclass", etc. In the developed system, only three types of CLIPS constructs are needed. Below is a brief description of each of these types along with how they are developed.

"deftemplate" constructs are used to define templates that create non-ordered facts (facts represent data in CLIPS) (CLIPS 6.10 1998a). The information required to develop these constructs for both domain entity types and their time-dependent set types consists of the definition of their attributes, attribute types, possible values, and default values. In each construct, every attribute is represented as a slot that has a name, type, allowed values or range, and default value. An additional slot called "case-type" is added to every construct to determine whether the fact generated from this construct belongs to the query case or the retrieved case. A complementary construct called "rule_firing_control" is added to avoid multiple firing of the same rule by storing the names of the previously fired rules.

"defrule" constructs are the heart of the knowledge base in CLIPS. They are used to define the collection of rules that describe how a specific problem is solved (CLIPS 6.10 1998a). The information required to develop these constructs for a specific domain entity type are encapsulated in the adaptation rules defined for that domain entity type as

presented in the previous section. Two complementary constructs are defined as starting and ending rules to open and close the output file in which the results will be stored.

"deffacts" constructs are used to define a collection of facts that are inserted automatically to the fact-list whenever these constructs are loaded (CLIPS 6.10 1998a). The primary source of these facts is the cases: the query case selected by the user as a primary input for the retrieval process and the retrieved case selected by the user for adaptation. Each case results in several facts: one fact represents the case domain entity and many facts represent its time-dependent sets.

For the developed system to interact with the CLIPS environment, "deftemplat", "defrule", and "deffacts" constructs are created for every individual domain entity type and stored in the text files named, respectively, "TypeName_temps.txt", "TypeName_rules.txt", and "TypeName_facts.txt" where "TypeName" is the name of the domain entity type. These files are then loaded into the CLIPS environment in the sequence shown in Figure 5.33 and manipulated by the CLIPS inference engine that provides the user with the adapted case. It should be noted that in case of having complex rules, these rules are defined in a text file called "TypeName_rules_additional.txt" that is appended to the rules defined in "TypeName_rules.txt" before loading it.

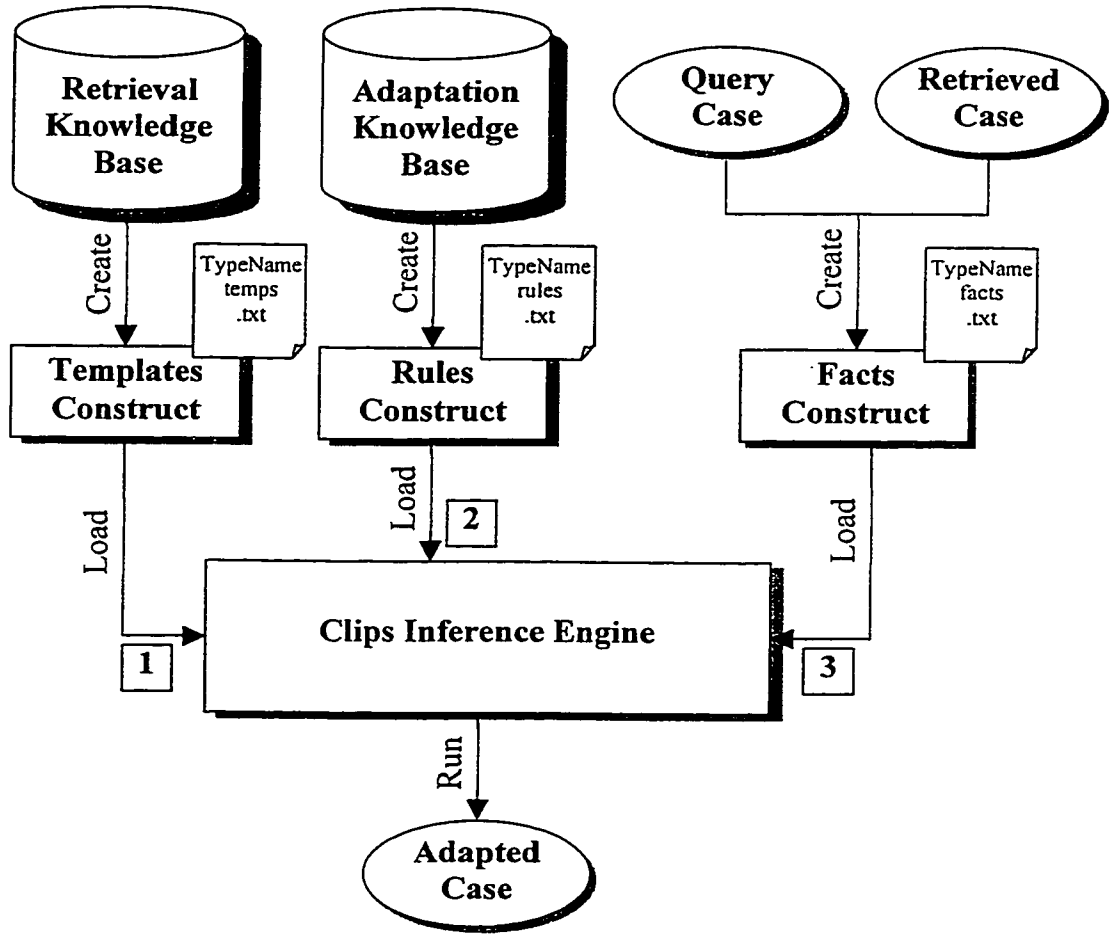


Figure 5.33: Processing adaptation rules in the developed system.

5.6 CASE ACCUMULATION AND MODIFICATION

A key issue in the success of any system design is how this system is maintained. Maintenance of CBR shells is defined as the execution of operations (called maintenance operations) that change the domain model, the case library, the retrieval knowledge base, and the adaptation knowledge base to eliminate any user errors or to reflect changes in reality (Heister and Wilke 1998). Maintenance operations in the developed CBR shell include case accumulation, knowledge accumulation, case modification, and knowledge modification as shown in Figure 5.34.

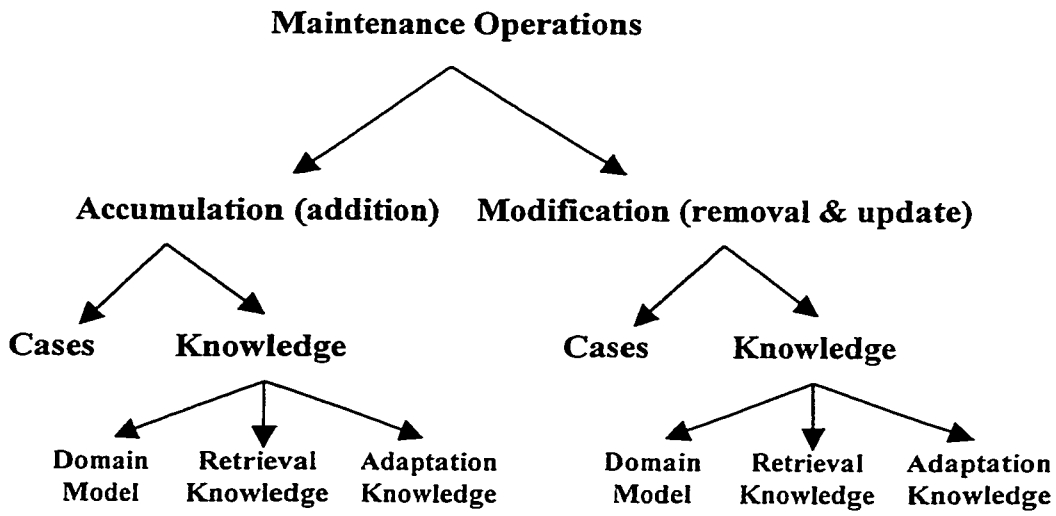


Figure 5.34: Types of system maintenance operations.

In the developed CBR shell, the System Maintainer module is responsible for all maintenance operations. This module acts as mediator between the domain expert or domain engineer, who initializes accumulation and modification transactions, and the OODBMS ObjectStore 6.0, which commits these transactions (refer to Figure 5.1). Figure 5.35 represents the generic algorithm used by the System Maintainer module in performing any maintenance operations. The above algorithm starts with opening the database that contains all persistent data. Then, it allows the user to initialize any allowed maintenance operation on this data such as add, remove, or update. Some maintenance operations are not allowed in certain situations, such as removing a domain entity type that has domain entities. After confirming the user selection, the algorithm continues by committing the database transaction that performs this operation. It also applies all the required repair operations, if any, and provides the user with the choice to do another maintenance operation or to exit and close the database. The main functions of the System Maintainer module can be summarized as follows:

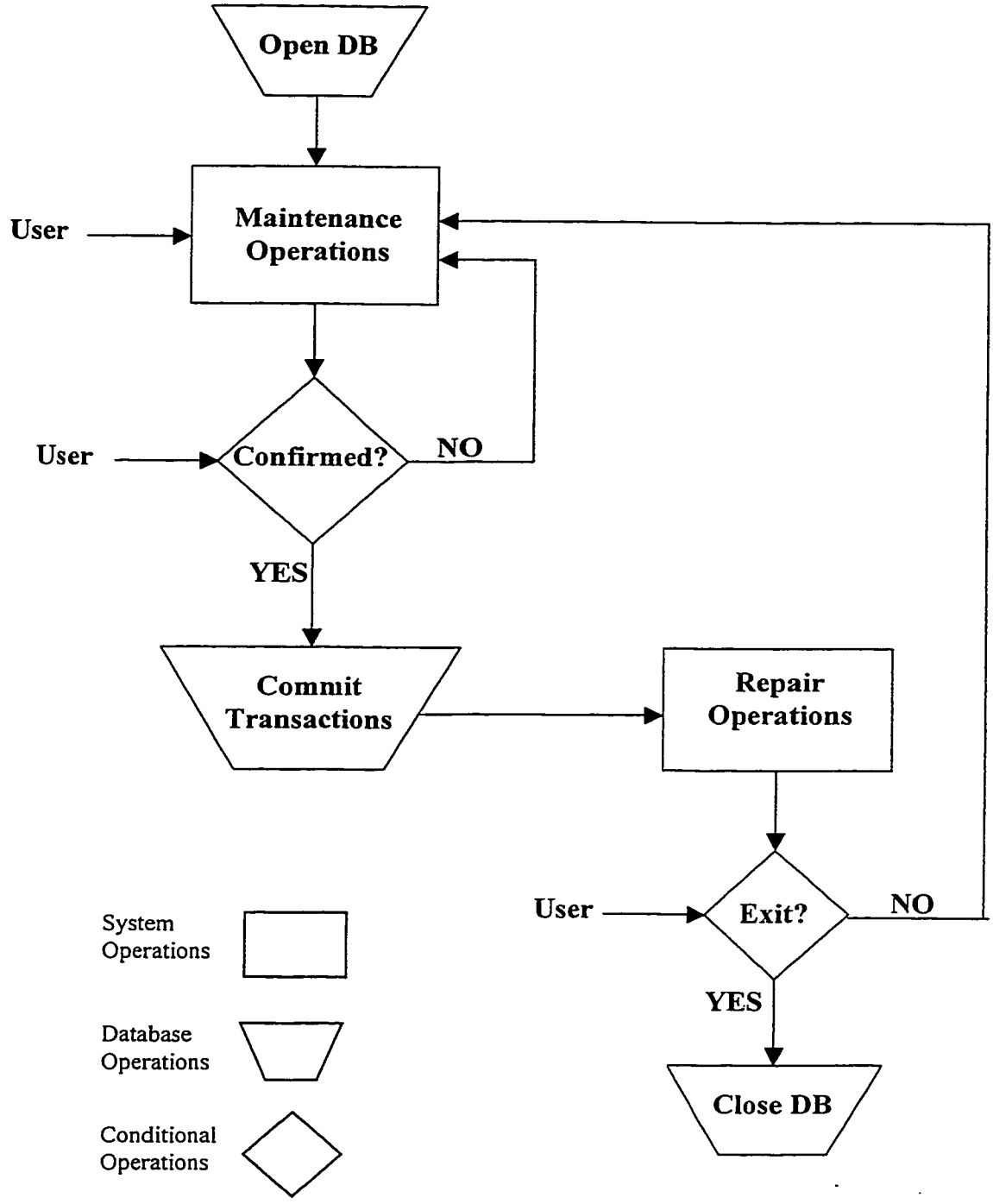


Figure 5.35: The algorithm used by the system maintainer

1. **Automating database operations:** The System Maintainer module instructs the OODBMS to perform some basic operations such as open database, commit transaction, save database, and close database with no need for user interventions. The objective of this function is to save the user time and effort spent in performing such traditional operations. These operations are represented in Figure 5.35 as trapezoidal shapes.
2. **Guiding the system user:** The user is responsible for initializing accumulation and modification transactions whenever they are needed. The System Maintainer module permits the initialization of legal transactions only and does not allow transactions that are not applicable or erroneous. It also validates the user inputs and asks for a confirmation before committing any of the legal transactions. A list of all transactions in the different system database sub-modules and when these transactions are not allowed is presented in Appendix C.
3. **Performing repair operations:** Some transactions may result in violating the referential integrity of the system database. Repair operations are performed after maintenance operations have taken place in order to keep the system database consistent. The System Maintainer ensures the referential integrity constraint in both deletion and addition transactions. In deletion transactions, repair operations are essential in order to avoid three types of problems: deleting wrong objects, referring to non-existing objects, and leaving unwanted objects without deletion. In addition transactions, none of the added object relationships will be established automatically. The System Maintainer module has to establish relationships according to the design model. Lists of the problems resulting from deletion and

addition transactions with different types of relationships and the corresponding repair operations taken by the System Maintainer module are presented in Appendix C.

CHAPTER 6

SYSTEM IMPLEMENTATION

6.1 INTRODUCTION

According to the system development cycle presented earlier in Chapter 3, the system design sub-phase of the build phase is followed by the system implementation sub-phase, which is sometimes called the system construction. In this sub-phase, the system design model is used as input to the code generation process. In object-oriented systems, this process is not simple because results obtained from the system design sub-phase are often incomplete. Several iterations and changes have to be made in order to construct a system that is easy to understand, extend, and maintain (Larman 1998).

In this chapter, system implementation issues such as the system resources, the implementation procedures, and the resulting software components will be discussed. A description of the developed prototype along with brief instructions for how to use it will be presented with an example.

6.2 IMPLEMENTATION ISSUES

The Rapid Database Development (RDD) tools of ObjectStore for C++ were utilized in generating the code that represents the implementation of the persistent classes defined earlier in the design model (Object Design 1998a). These tools include the Database Designer and the Component Wizard. The Database Designer is a tool for designing ObjectStore databases. It provides a graphical user interface (GUI) that allows the

creation and maintenance of database elements (i. e. classes, data members, relationships, and member functions) in a quick and efficient manner. Using the Database Designer and the design model developed in Chapter 5, a database design was defined. The Component Wizard is an application wizard for VC++ 6.0. This wizard uses the database design to generate ready-to-use ObjectStore components for C++. These components include (Object Design 1999): header and source files for every class in the database design along with coding for create, read, update and delete functions; an ObjectStore schema source file that is used as input to the Object Store schema generator; a Dynamic Link Library (DLL) definition file that contains export comments for functions needed to create persistent objects; a DLL project file that is used to build the ObjectStore DLL containing schema, declarations, and definitions for every element in the database design; and workspace file that bundles all custom projects that work with the database project.

In order to satisfy the system modularity requirement, seven separate executable (EXE) projects were developed in the same workspace as the ObjectStore database project using the object-oriented programming (OOP) language C++. The Integrated Development Environment (IDE) of Microsoft Visual Studio 6.0 and the Microsoft Foundation Classes (MFC) were utilized in developing these projects. Although each project has its own code that performs a specific functionality, these projects are dependent either on each other or on the database project. These projects were compiled, using the VC++ 6.0 compiler, according to their dependencies (independent projects first and dependent projects after). Then, they were linked to provide the system with seven software components: "Main.exe", "Domain_Model.exe", "Retrieval_Knowledge.exe",

"Case_Library.exe", "Adaptation_Knowledge.exe", "Template_Library.exe", and "Retrieval_&_Adaptation.exe". Figure 6.1 shows the process diagram that represents all software components of the developed system and their dependencies.

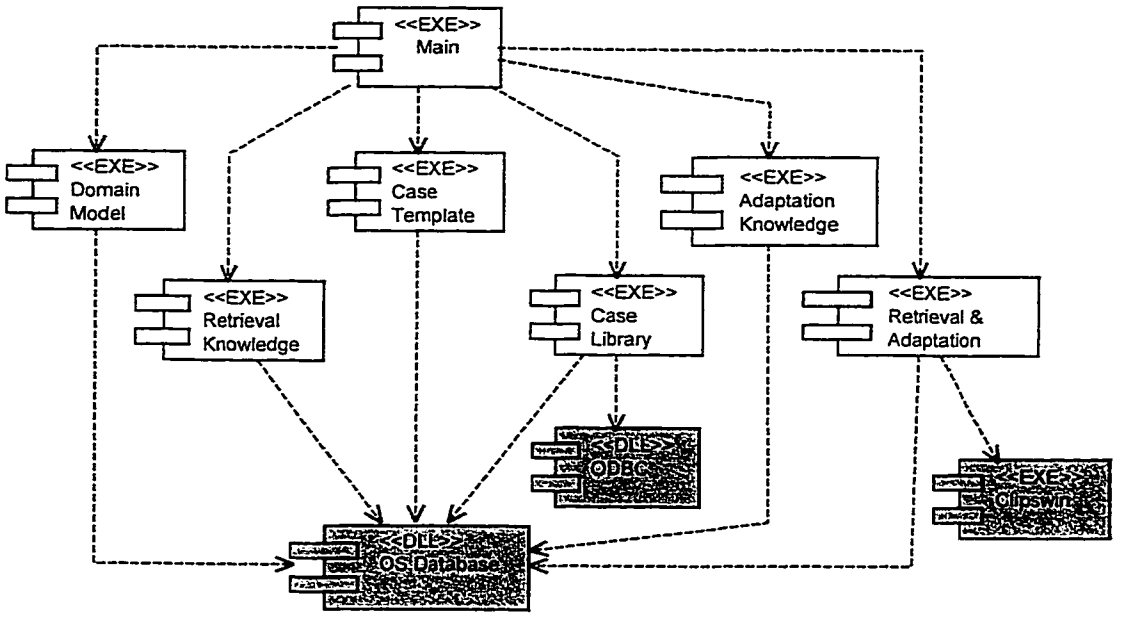


Figure 6.1 Process diagram of the developed system

In this diagram, three components (shown in shaded rectangles) were not developed by the author, but are ready-made components that were reused in the developed system. These components are "Clipswin.exe", "ODBC.dll", and "OS_Database.dll". "Clipswin.exe" is an external component that was not included in the developed workspace. This component is needed to run the inference engine of CLIPS 6.0 that is used by the "Retrieval_&_Adaptation.exe" component to provide the system with case adaptation capabilities. "ODBC.dll" is a component that is made by Leinecker and Archer (1998) and was included in the developed workspace to be used by the "Case_Library.exe" component to provide the system with data transfer capabilities. This component allows the case library to import data from any Object Database Connectivity

(ODBC) data sources such as Microsoft Access databases, dBASE files, FoxPro Files, and Excel files. "OS_Database.dll" is a component generated by the Component Wizard and was included in the developed workspace to be used by all the components that need to access the database for read or write operations. It should be noted that all the developed software components, except the "Main.exe", need to access the database component for both read and write operations. They also correspond to the CBR shell modules and sub-modules presented earlier in Figure 4.1. This correspondence helps in identifying the intended functionality of each component. The "Main.exe" is the component responsible for the system login, which verifies user authorization; and the system main menu, which provides the user with the accessible system modules.

In order to obtain an understandable and maintainable coding, an efficient strategy was used in generating the source code of each of the developed components. In this strategy, each component is composed of three types of object-oriented classes: control, boundary, and entity classes. Only one control class called "Manager" was created in every component. By definition, a control class is the class that coordinates the flow of events needed to perform a use case (Quatrani 1998). One object from the "Manager" class is constructed every time a component starts and is destructed whenever it ends. This object has a one-to-many bi-directional aggregation relationship with objects instantiated from boundary classes. By definition, a boundary class is the class that provides the interface to a user or to another system. The "CDialog" class from MFC was used as a base class for all boundary classes created in the developed system to build its GUI. The object instantiated from the "Manager" class also has a one-to-one unidirectional aggregation

relationship with an object instantiated from a class called "OS_Database". This object represents one ObjectStore database that contains many objects instantiated from different entity classes. By definition, an entity class is the class that models long-lived information (i.e. all persistent classes are entity classes).

Figure 6.2 depicts a simplified class diagram of the strategy used. The main advantage of such a strategy is that it separates the code used to manipulate system data from the code used to build the system GUI. This separation localizes modifications and simplifies system debugging. A detailed class diagram for each of the developed components is presented in Appendix D.

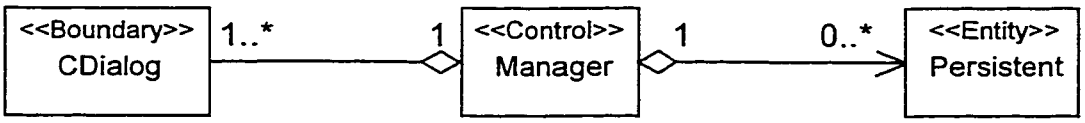


Figure 6.2 Simplified class diagram for the system components.

In order to clarify the effectiveness of the strategy described above and the importance of the modularity requirement in the implementation of complex software systems, Table 6.1 presents the total number of code lines used in implementing the proposed system. This table shows how the heavy coding burden was distributed over the three types of classes and the seven separate projects. This distribution aided the system developer in modifying and debugging the generated code easily and in a timely fashion.

Total Number of Code Lines					23106
	Entity Classes			5940	
	Control Classes			7864	
		Main	163		
		Domain Model	513		
		Retrieval Knowledge	1765		
		Adaptation Knowledge	926		
		Case Template	406		
		Case Library	2119		
		Retrieval & Adaptation	1972		
	Boundary Classes			9302	
		Main	437		
		Domain Model	650		
		Retrieval Knowledge	3285		
		Adaptation Knowledge	891		
		Case Template	612		
		Case Library	2460		
		Retrieval & Adaptation	967		

Table 6.1: Distribution of the code lines over the system classes and components

6.3 PROTOTYPE DESCRIPTION

The developed software system is a prototype of a CBR shell that can be used in modeling the deterioration of infrastructure facilities (e.g. bridges, pavements, culverts, roofs, etc.). This prototype is called CBRMID, which stands for Case-Based Reasoning for Modeling Infrastructure Deterioration. This prototype works as a 32-bit application for Windows 95 or later platforms. In developing this prototype, considerable attention was given to the user interface due to the fact that efficient and convenient communication between a system and its users is crucial especially in highly data-intensive systems like this one. Also, a consistent and user friendly GUI limits data entry errors and avoids confusions and misunderstandings.

Figure 6.3 shows the "Login" dialog that appears once the user runs the "Main.exe" component. This dialog allows the three actors, defined earlier in the Use Case diagram presented in Chapter 3, to access the different system modules. These actors are: a domain expert, who has the authority to access all system modules; a domain engineer, who has the authority to access the "Case Template", "Case Library", and "Retrieval and Adaptation" modules; and a decision maker or a IMS module that can only consult the system through accessing the "Retrieval and Adaptation" module. In order to login, the user has to select an actor that represents his/her authorization. Both domain expert and domain engineer actors have to enter the appropriate password in order to have their privileges, while the decision maker or the IMS module can access the system without a password.

Figure 6.3 Login dialog

Figure 6.4 shows the "Main Menu" dialog in the case of a domain expert actor. This dialog contains six different system modules ordered according to the proper sequence that should be followed. This dialog allows the user to create any number of databases. Each database has a unique name and can be accessed by a single user for both read and write transactions. Only one system module can access the database at a time and any transaction made in that module is committed immediately to the database as long as the transaction is ended safely. In the following sub-sections, each system module will be described in detail using an example.

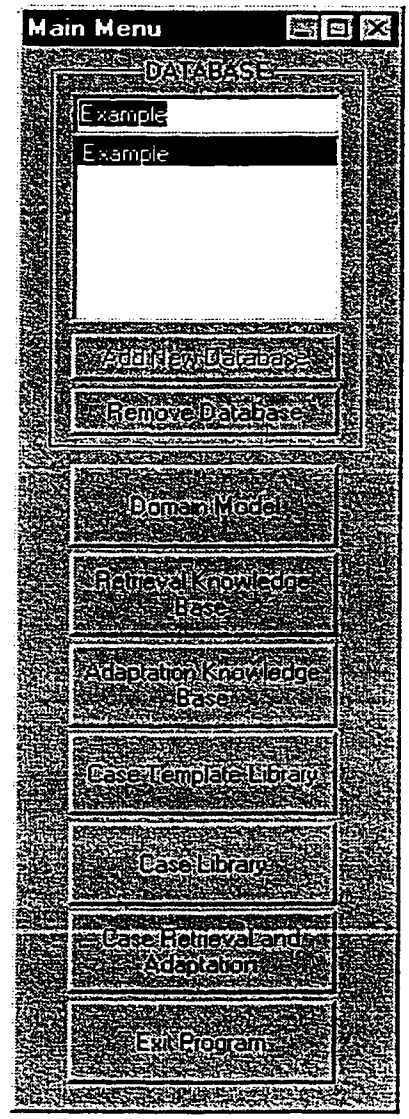


Figure 6.4 Main Menu dialog

6.3.1 DOMAIN MODEL MODULE

In order to use the developed CBRMID in modeling the deterioration of an infrastucture facility, a domain model that defines domain concepts and their relationships has to be built first. This is done by using the "Domain Model" module in CBRMID. The conceptual model of bridges presented in Chapter 4 is a good example of a domain model that represents highway bridges. To build this model, domain entity types are created, as shown in Figure 6.5, and arranged in a hierarchical decomposition that reflects their aggregation relationships presented in Figures 4.11 to 4.15.

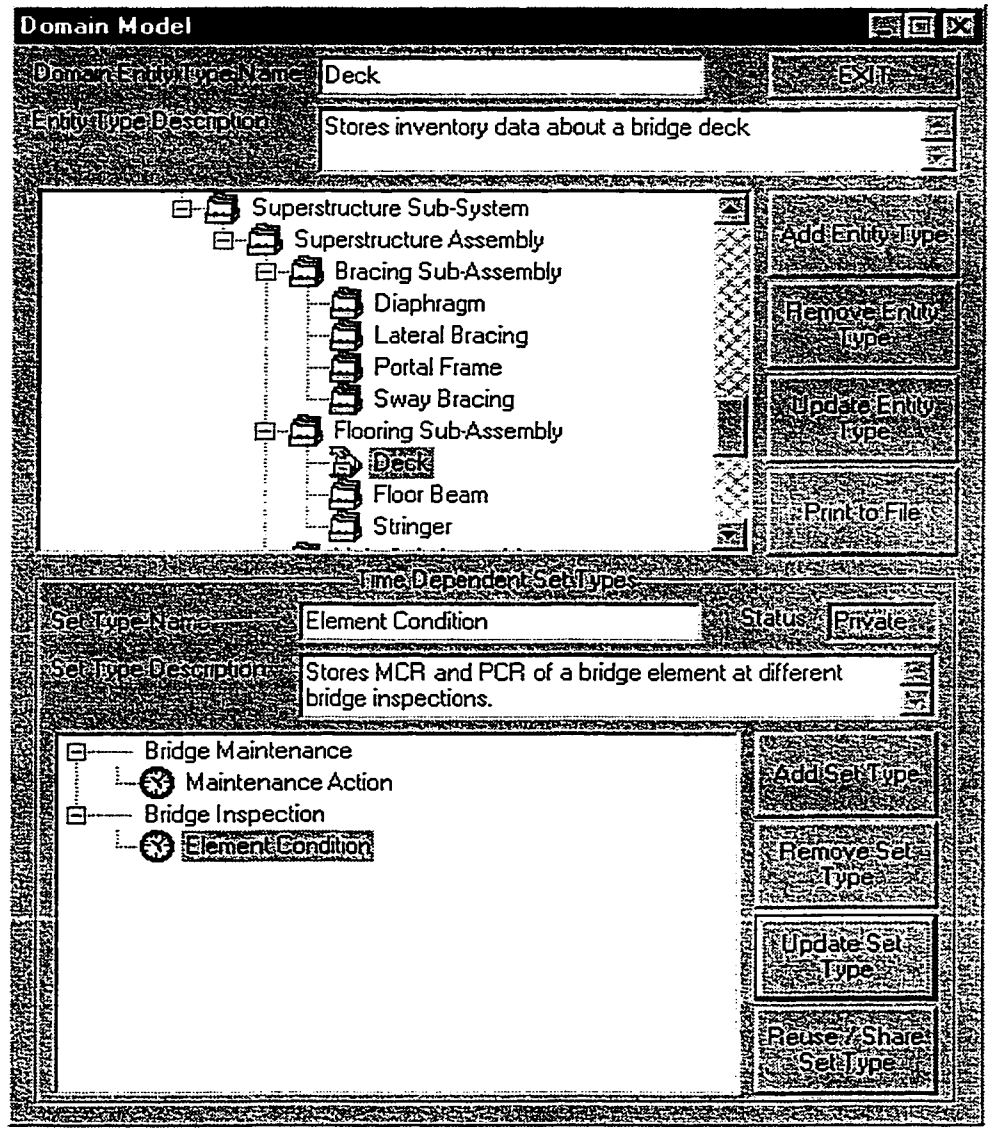


Figure 6.5: Domain Model dialog

When creating a new domain entity type (e.g. Deck), the user has to select the parent domain entity type (e.g. Flooring Sub-Assembly) and enter the name of the entity type, which has to be a non-null and unique name. A narrative description of the domain entity type can also be added. Both entity type name and description can be updated.

Some domain entity types have data that are time-dependent such as inspection and maintenance data. To store this data, time-dependent set types such as "Bridge Inspection", "Element Condition", "Bridge Maintenance", and "Maintenance Action" have to be created. To create a time-dependent set type (e.g. Element Condition), the user has to select a parent domain entity type (e.g. Deck), select a parent time-dependent set type (e.g. Bridge Inspection) if any, and enter a non-null and unique set type name. Time-dependent set types that belong to a specific domain entity type are designated by a watch image on their left when this entity type is selected (see Figure 6.5).

Although each time-dependent set type must be contained in a domain entity type, it can be reused (i.e. contained in several domain entity types) or shared (i.e. its time-dependent sets are contained in several domain entities). For instance, the time-dependent set type "Element Condition" is reused by many bridge elements such as "Wall", "Column", "Cap", "Girder", etc. To do this, the user has to select a domain entity type (e.g. Girder), press the "Reuse/Share Set Type" button to activate the dialog shown in Figure 6.6, select the "Element Condition" time-dependent set type, and press the "Reuse Set Type" button. The same procedures are followed in sharing a set type, provided that reused set types cannot be shared and vice versa. The status of any set type (i.e. Reused,

(i.e. Reused, Shared, Private, or None) with respect to its parent domain entity type is shown in the "Status" edit box shown in Figure 6.5.

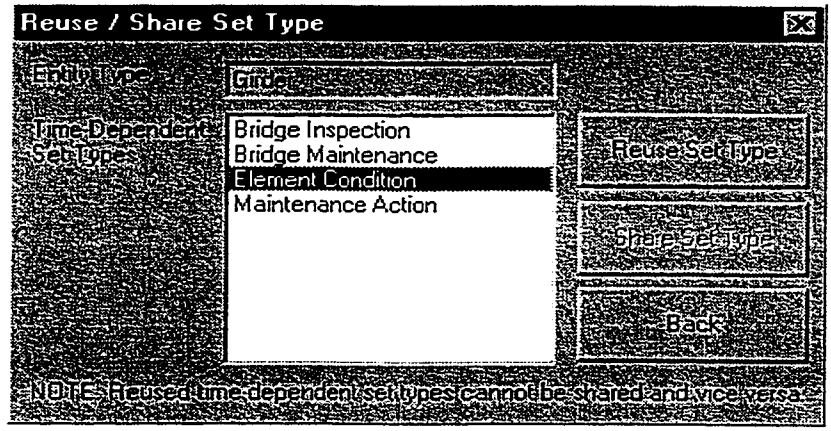


Figure 6.6: Reuse and Share Set Type dialog

Removing a reused or shared time-dependent set type does not delete this set type but just removes its relationship with the selected domain entity type. In order to remove a domain entity type or a private time-dependent set type completely from the database, any related objects (child types, attributes, adaptation rules, entities, sets, etc.) must be deleted first. This restriction avoids the accidental removal of objects and provides more secure transactions. In any of the system modules, the "Add", "Remove", and "Update" buttons are used for committing, respectively, add, remove, and update (ARU) transactions after the user confirmation. Also, the "Print to File" button is used to document the module information in a text file, the "Back" button" is used to return to the previous dialog, while the "Exit" button is used to return to the main menu.

6.3.2 RETRIEVAL KNOWLEDGE MODULE

Domain entity types and time-dependent set types defined in the previous module are empty containers that need to be filled with attributes. These attributes are the data items

that describe case contents. Attributes of both domain entity types and time-dependent set types are defined by using the "Retrieval Knowledge" module in CBRMID. In this module, the user can define all the attribute properties such as attribute name, default value, group, description, type, role, weight, and similarity measure along with aggregation weights of parent domain entity types to be used in matching sub-cases. Figure 6.7 shows the attributes defined for the "Deck" domain entity type. It also shows the properties of the "Skew Angle" attribute, which is a discriminating attribute of a numeric type with continous values, and a weight equal to 0.2 (i.e. slightly important).

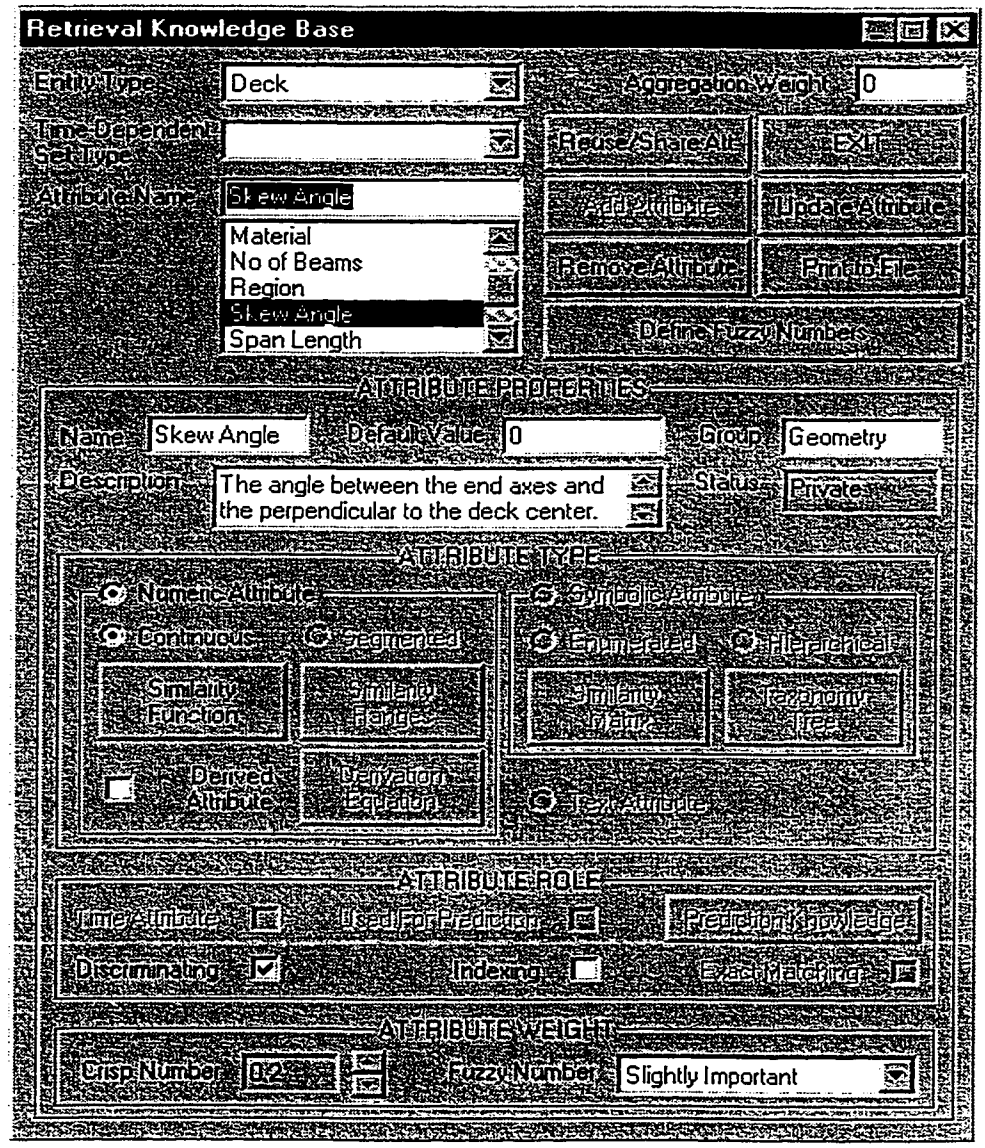


Figure 6.7: Retrieval Knowledge dialog

In order to determine the similarity among the values of an attribute, four different similarity measures are provided for different types of attribute. Figure 6.8 shows the "Similarity Function" dialog used with numeric continuous attributes (e.g. Skew Angle). In this dialog, the user has to define some properties that belong to numeric attributes only, such as unit, upper limit, lower limit, and whether the attribute is a date attribute or not. Also, the user has to determine the parameters of the similarity function: the function pattern, function intervals, whether the function is symmetrical or not, and whether the difference is relative or absolute. For more details about these parameters, refer to section

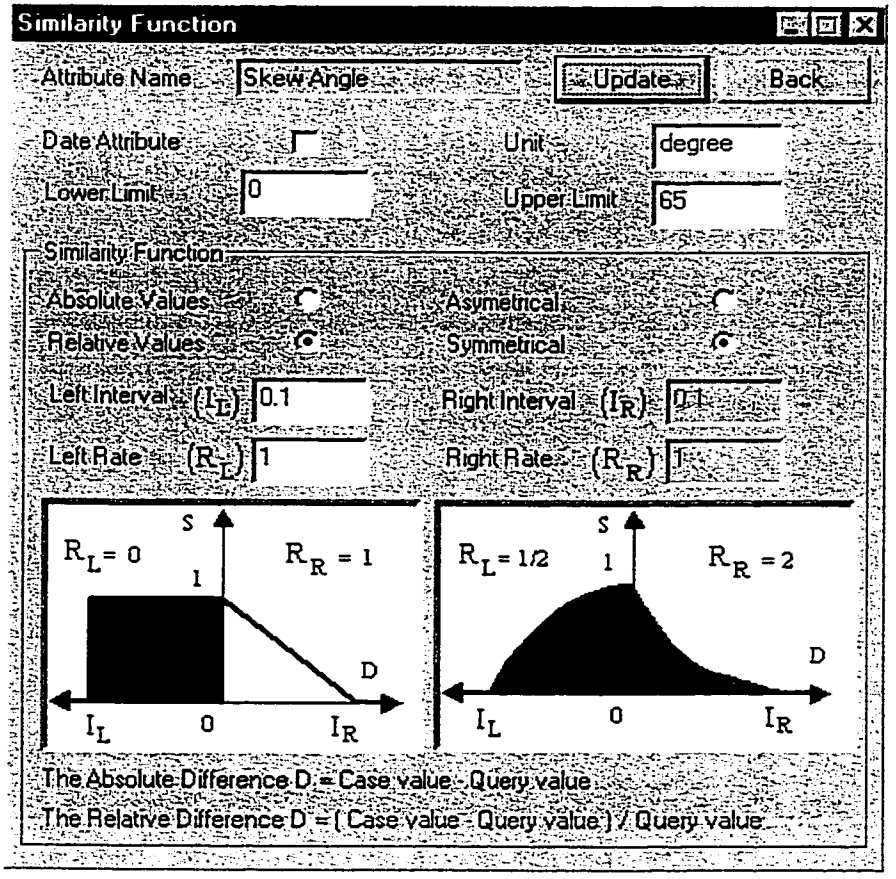


Figure 6.8: Similarity Function dialog

5.4.1.1.

For numeric segmented attributes (e.g. Span Length), the "Similarity Ranges" dialog shown in Figure 6.9 is used. In this dialog, the user defines the additional properties of the numeric attribute along with its similarity ranges. Each range is determined by two numeric values (absolute or relative) and a degree of similarity that can be expressed as a crisp number or as a fuzzy number.

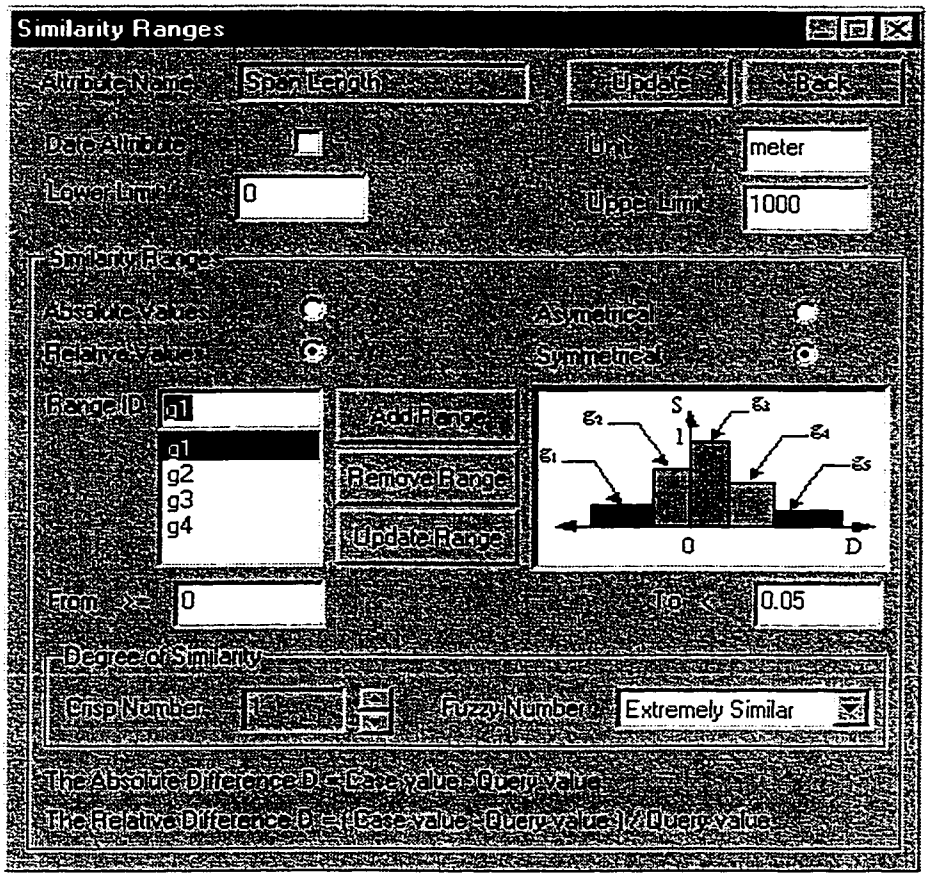


Figure 6.9: Similarity Ranges dialog

For symbolic attributes with enumerated values (e.g. Region), the "Similarity Matrix" dialog shown in Figure 6.10 is used. In this dialog, each cell in the similarity matrix represents the degree of similarity (crisp or fuzzy number) between a pair of attribute values. Because of the symmetry of similarity matrices, only the lower or the upper triangle of a similarity matrix needs to be filled.

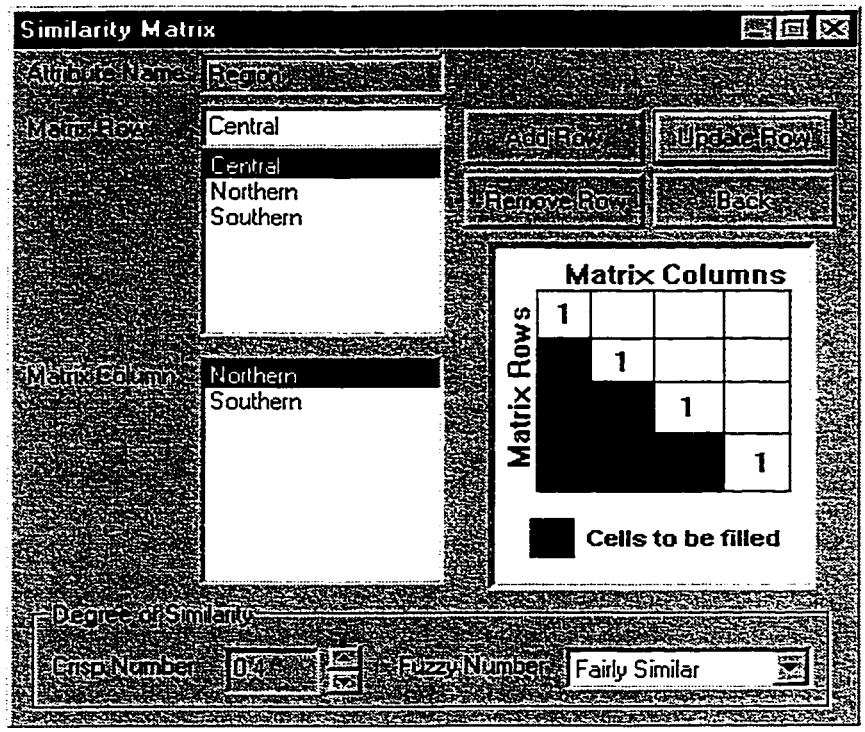


Figure 6.10: Similarity Matrix dialog

For symbolic attributes with hierarchical values (e.g. Material), the "Taxonomy Tree" dialog shown in Figure 6.11 is used. In this dialog, all possible values of a symbolic attribute are represented in a taxonomy tree using different decomposition criteria. Each parent node in this tree (e.g. Concrete) has a criterion (e.g. Casting Method) on which the decomposition is carried out. It also has a degree of similarity (crisp or fuzzy number) among its child nodes (e.g. Cast-in-Place and Precast). It should be noted that the value of a hierarchical attribute at any node in a taxonomy tree is represented by the whole path from the tree root to this node. The value of a hierarchical attribute, its degree of similarity, and its criteria can all be updated.

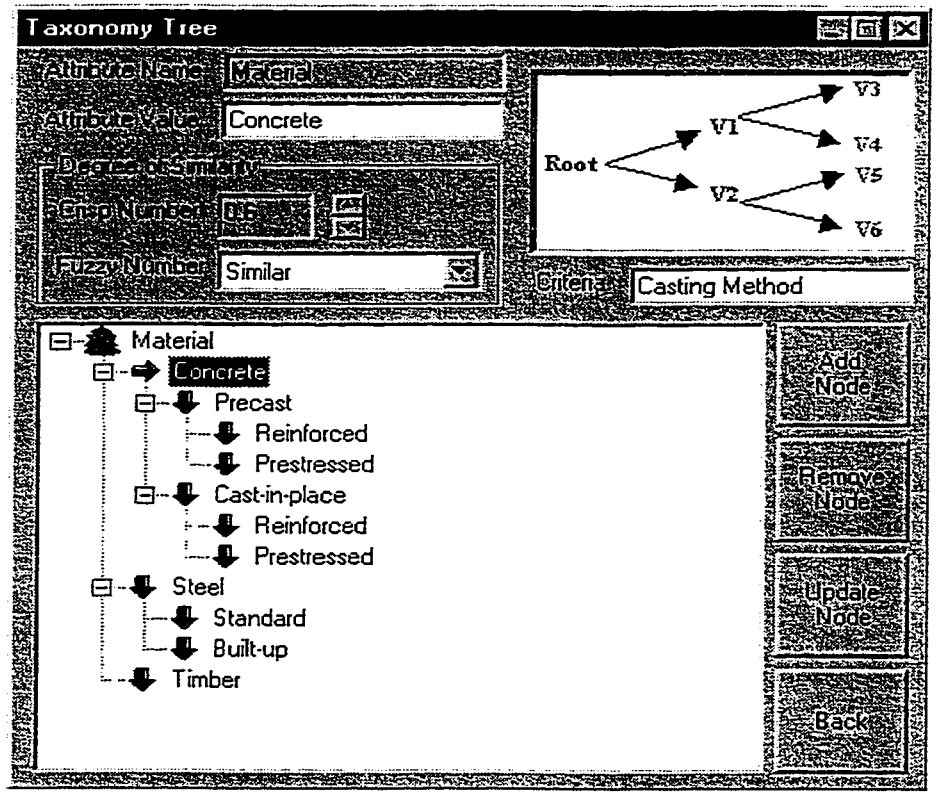


Figure 6.11: Taxonomy Tree dialog

Some attributes (e.g. Beam Spacing) can be derived from other attributes (e.g. Total Width and No. of Beams). The derivation equation is defined using the dialog shown in Figure 6.12. In this dialog, the user has to select the source or location of the attributes used in deriving a new attribute and build the equation as a set of polynomial terms with a relationship among terms and a derivation constant. For deriving the "Beam Spacing" attribute, attributes from the same container are used as term attributes with a multiplication relationship among their polynomial terms and with a derivation constant equal to 1. The first term contains the "Total Width" as a term attribute with a coefficient equal to 1 and a power equal to 1. The second term contains the "No. of Beams" as a term attribute with a coefficient equal to 1 and a power equal to -1 to convert the multiplication into division.

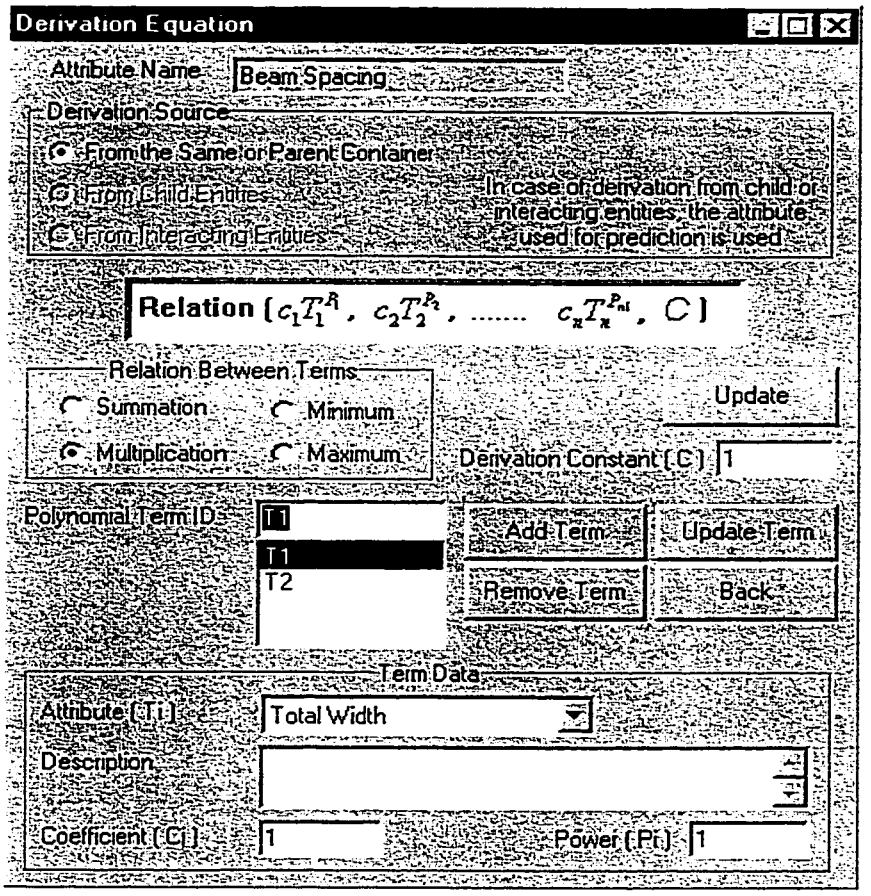


Figure 6.12: Derivation Equation dialog

Some attributes (e.g. Material, Structural System, etc.) can be used to describe several domain entity types (e.g. Column, Girder, Deck, etc.), but defining these attributes for every individual domain entity type is redundant. Reusing pre-defined attributes in different domain entity types or time-dependent set types can be done using the dialog shown in Figure 6.13. This dialog can also be used in sharing attributes (i.e. reusing attributes and their values) that belong to a parent entity type or set type . For example, the "Region" attribute defined for the "Bridge" domain entity type can be shared by other domain entity types (e.g. Girder). This means that the value of the "Region" attribute in all girder entities is the same as the value of the same attribute in the parent bridge entity.

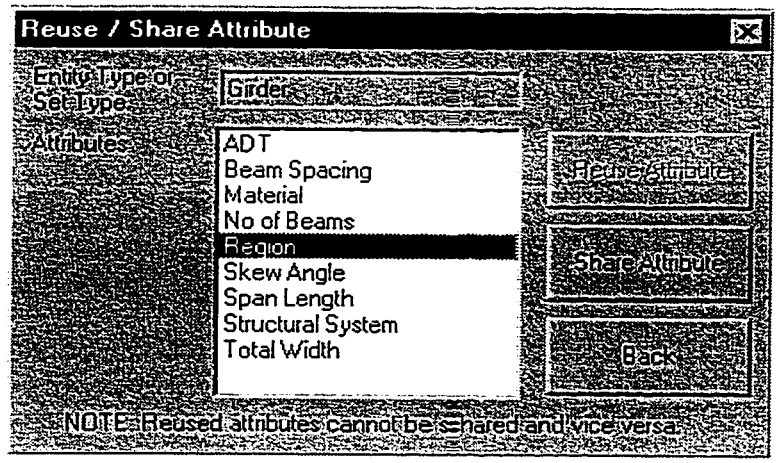


Figure 6.13: Reuse and Share Attribute dialog

In estimating weights of attributes and degrees of similarities among their values, fuzzy numbers are used to capture the subjectivity of human estimation. These fuzzy numbers are defined using the dialog shown in Figure 6.14. Every fuzzy number has a non-null and unique name, a description, and three crisp values representing its triangular shape. Also, the variable (e.g. degree of similarity or degree of importance) that this fuzzy number belongs to has to be determined.

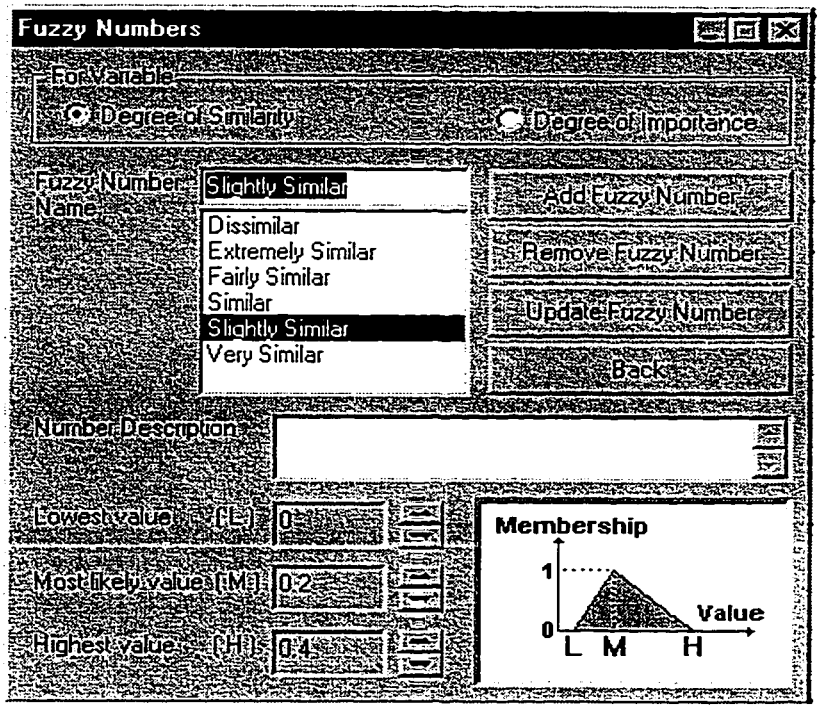


Figure 6.14: Fuzzy Numbers dialog

Attributes that belong to time-dependent set types (e.g. Element Condition) have additional properties, such as being a time attribute (e.g. Inspection Date) or an attribute used for prediction (e.g. MCR). Figure 6.15 shows the "Prediction Knowledge" dialog used to define the properties required for attributes used for prediction such as: Maximum number of time points needed for providing a satisfactory comparison of time series, maximum attribute weight that is achieved whenever a full comparison is carried out, and a time-weight factor that changes the weight of matched points according to how old they are. These properties have to be defined by domain experts according to their experience and needs.

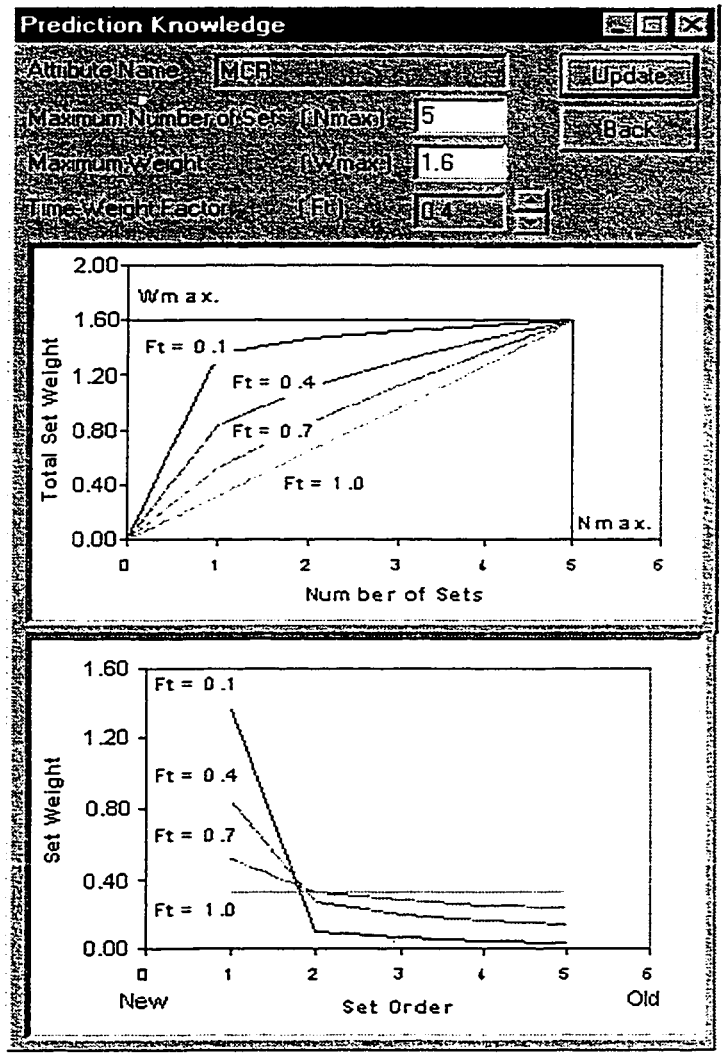


Figure 6.15: Prediction Knowledge dialog

6.3.3 ADAPTATION KNOWLEDGE MODULE

After defining the knowledge required for the retrieval process, domain experts can define the knowledge required for case adaptation. This knowledge is expressed in the form of adaptation rules that are created for every domain entity type using the "Adaptation Knowledge" module in CBRMID. In this module, adaptation rules can be stored either as objects defined in the adaptation knowledge base or as CLIPS rules written in a text file. Figure 6.16 shows the dialog used in formulating an adaptation rule "R1" for the "Deck" domain entity type. This rule adapts deck condition to account for the difference in its structural system between the query and retrieved cases. Each adaptation rule has a non-null and unique identifier, a description, and a salience that can be used to control the sequence of rule firing. Each rule can be marked as a hidden rule and as a rule that can be fired only once. The "Additional Rules" button shown in Figure 6.16 is used to open an editor that allows the user to write additional adaptation rules using CLIPS syntax.

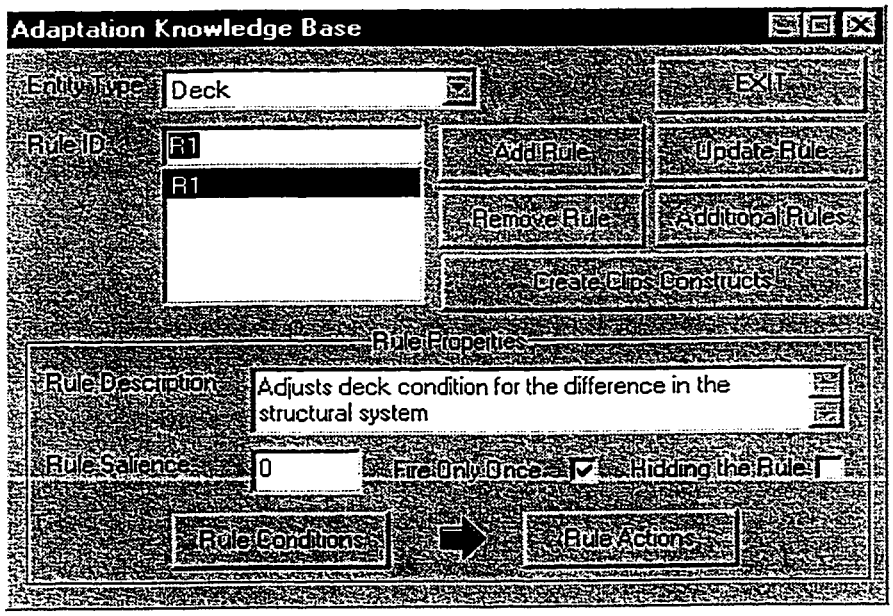


Figure 6.16: Adaptation Knowledge dialog

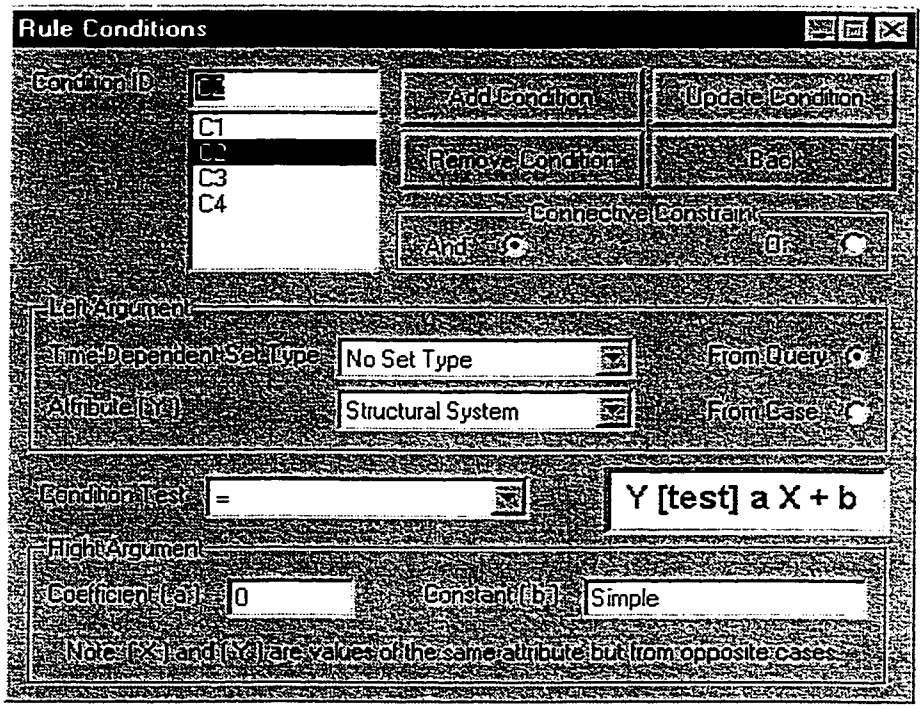


Figure 6.17: Rule Conditions dialog

Figure 6.17 shows the dialog used for formulating the conditions of the rule "R1" (i.e. left-hand side). This dialog shows four conditions named "C1", "C2", "C3", and "C4". All these conditions are connected with the "And" connective constraint, which means that the four conditions must be satisfied for the rule to fire. The first condition shown in Figure 6.17 is satisfied when the structural system in the query case is not the same as in the retrieved case. The second condition (the one shown) is satisfied when the structural system in the query case is "Simple". The third and fourth conditions are satisfied when the difference in the span length between the query and retieved cases is within ±10%.

Figure 6.18 shows the dialog used for formulating the actions of the rule "R1" (i.e. right-hand side). A rule action may assign a value for the retrieved case (i.e. automatic adaptation) or may provide some instructions to the user (i.e. manual adaptation). The

rule action shown in Figure 6.18 modifies the retrieved deck by subtracting 0.5 from the value of its material condition rating to compensate for the difference in the value of the "Structural System" attribute.

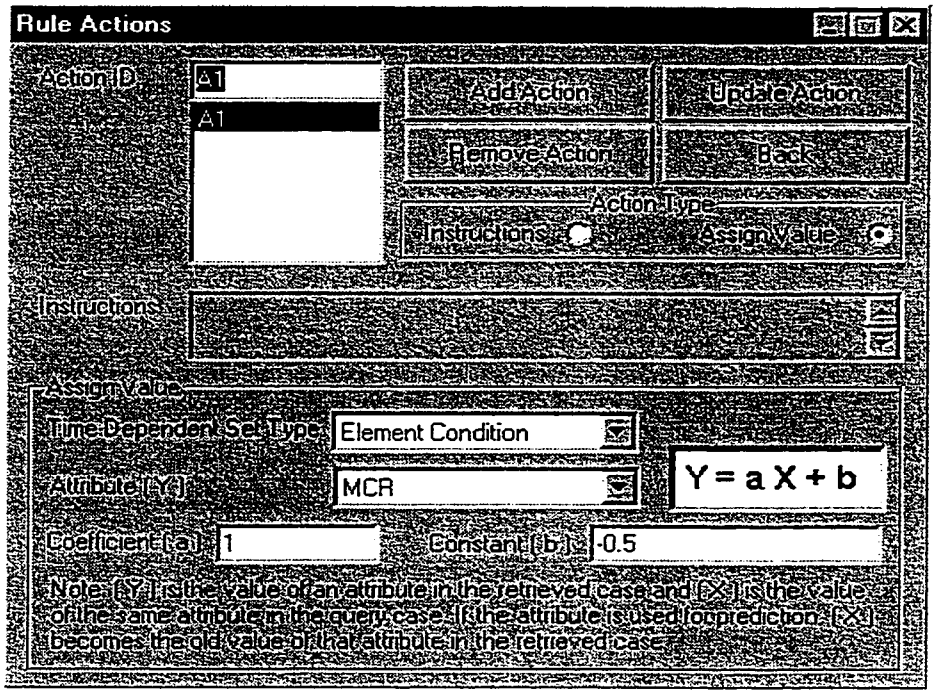


Figure 6.18: Rule Actions dialog

After defining the adaptation rules, CLIPS constructs (i.e. "deftemplate" and "defrule" constructs) are created once the user presses the "Create Clips Constructs" button shown in Figure 6.16. This option creates two text files: "Deck_temps.txt" that contains the "deftemplate" constructs for the "Deck" domain entity type; and "Deck_rules.txt" that contains the "defrule" constructs for the rules stored in the adaptation knowledge base and the additional rules stored in the text file "Deck_rules_additional.txt" of the same domain entity type.

6.3.4 CASE-TEMPLATE LIBRARY MODULE

Before adding cases to the case library, the user can create case templates that make the process of storing cases simpler and quicker. These templates are created using the "Case Template Library" module in CBRMID. Figure 6.19 shows the dialog used in creating three templates for three different types of bridges using the domain entity types previously defined in the domain model. The template "Type01" represents the hierarchical decomposition of a two-span girder bridge.

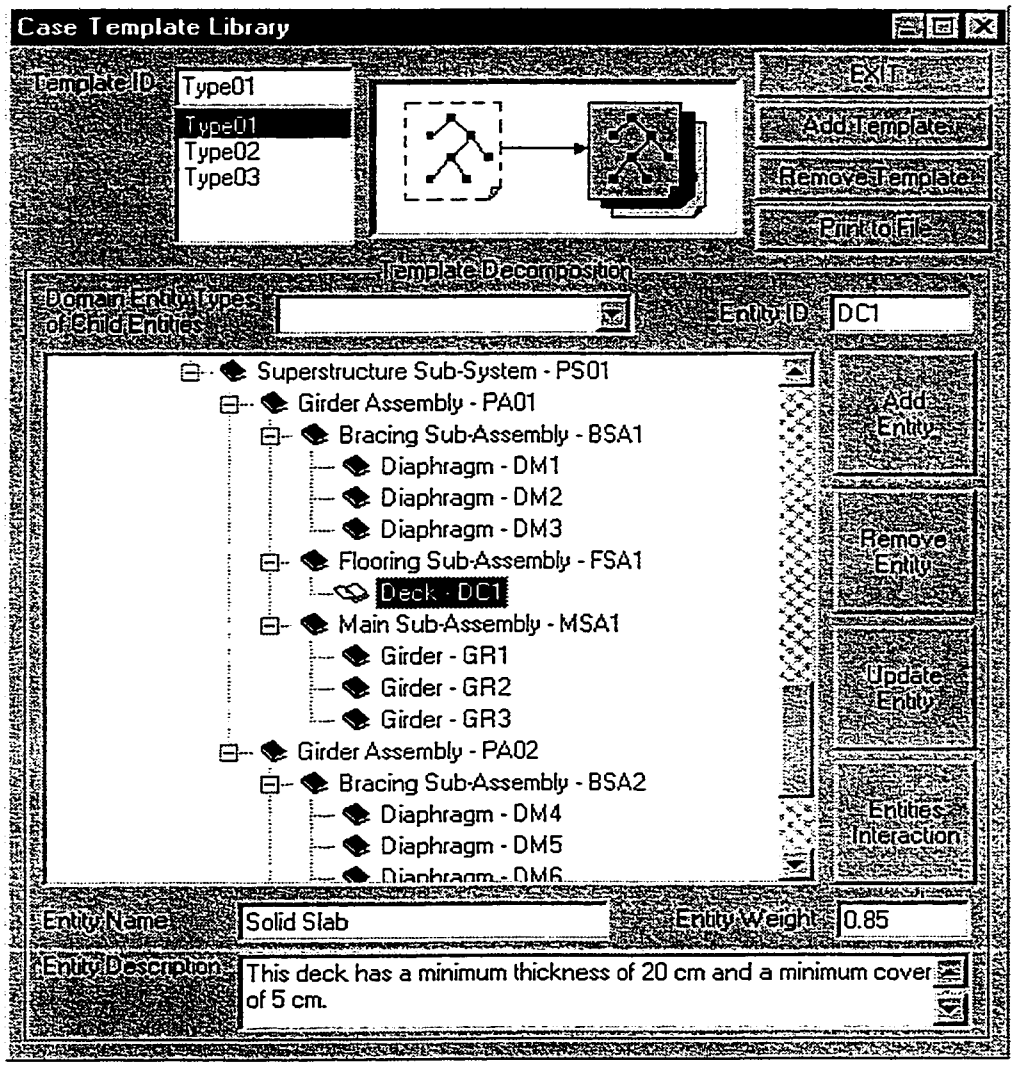


Figure 6.19: Case Template Library dialog

To add a domain entity in a case template, the user has to select the parent domain entity, specify the domain entity type, and enter a non-null and unique entity identifier. Entity description, name and weight can also be added. In order to define the entities interacting with a specific domain entity (e.g. Deck DC1), the dialog shown in Figure 6.20 is used. This dialog lists all the available domain entities that belong to a domain entity type selected by the user (e.g. Girder). The three girders "GR1", "GR2", and "GR3" which support the deck "DC1" have been selected as its interacting entities. The condition of these girders will be considered in predicting the future condition of that deck. The interaction weight of each girder has to be specified before the user presses the "Add >>" button. These weights represent the participation of interacting entities in the interaction mechanism. To remove an entity from the list of interacting entities, the user has to select this entity and press the "<<Remove" button.

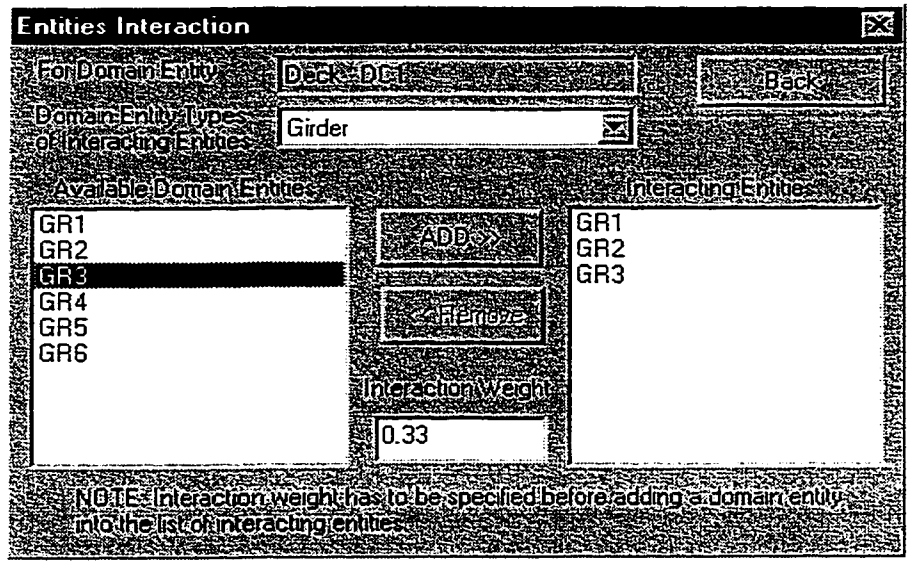


Figure 6.20: Entities Interaction dialog

6.3.5 CASE LIBRARY MODULE

In the case library module, case templates defined in the previous module are utilized in creating new cases. To create a new case, the user has to enter a non-null and unique case identifier (e.g. B01), select an appropriate case template (e.g. Type01), and press the "Add Case" button". The resulting case shown in Figure 6.21 has the same structure as the corresponding template and the same id, name, descriptions and weight of all entities. Selecting a domain entity in the hierarchical decomposition of cases activates several options in the case library dialog. Next, is a discussion of each of these options.

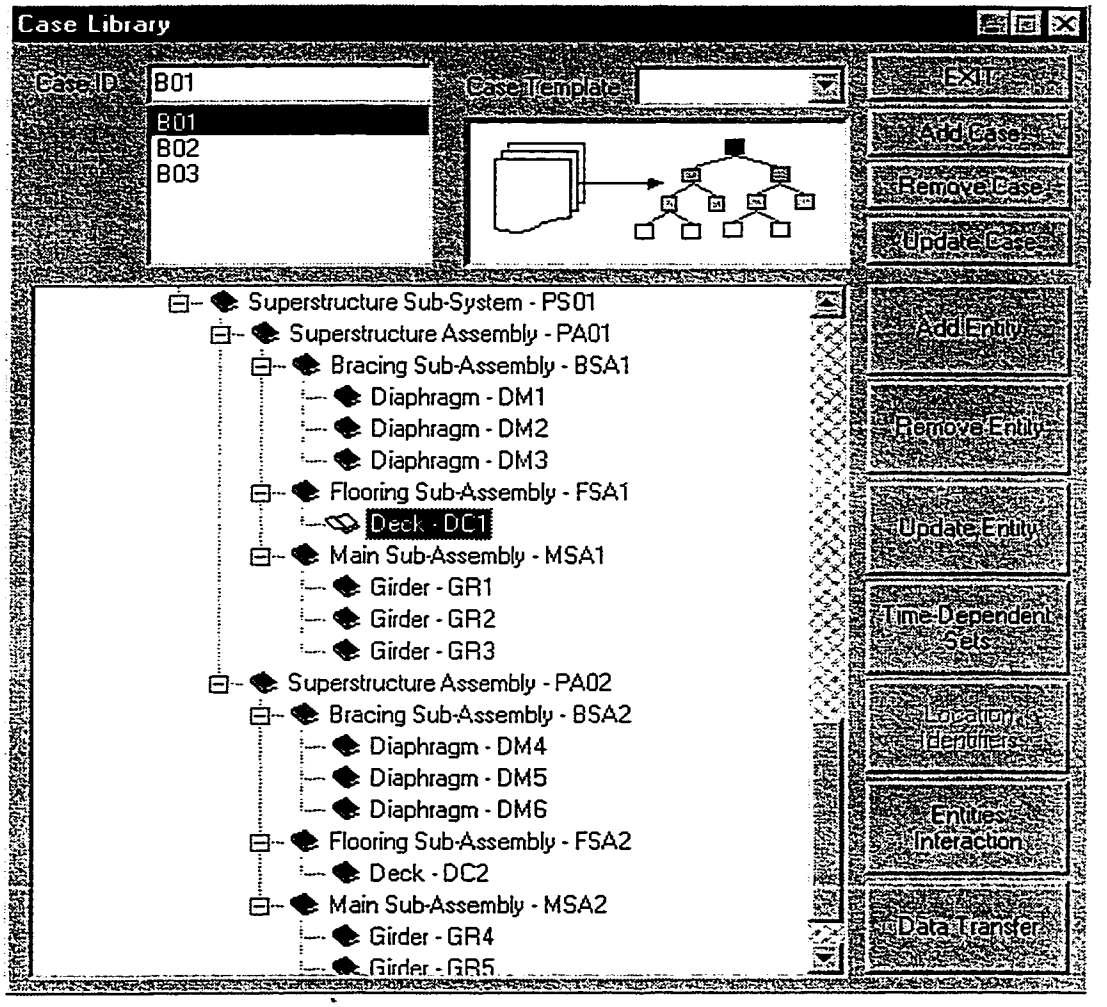


Figure 6.21: Case Library dialog

Quite often, cases are not exactly the same as the templates used for creating them. Some domain entities need to be added while others need to be removed. In order to add a new domain entity (e.g. a girder) in the hierarchical decomposition of the case "B01" shown in Figure 6.21, the user has to select the parent domain entity (e.g. Main Sub-Assembly MSA2). Then, once the user presses the "Add Entity" button, the "Add Entity" dialog shown in Figure 6.22 appears. In this dialog, the user has to identify the domain entity type (e.g. Girder) of the new entity, the entity name (e.g. Precast T Girder), and the entity id (e.g. GR7). This dialog is also used to create new cases from scratch whenever appropriate case templates are not available. To remove a domain entity, the user has to select this entity and press the "Remove Entity" button shown in Figure 6.21.

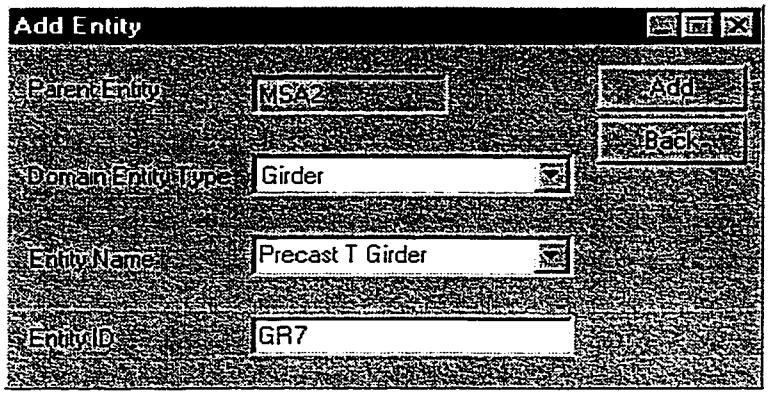


Figure 6.22: Add Entity dialog

Data that describe case contents has to be stored in every individual domain entity in the case hierarchical decomposition. These data are classified into: static data and dynamic data (i.e. time-dependent data) Figure 6.23 shows the "Update Entity" dialog that is used to store and update all static data of a specific domain entity. This dialog appears once the user selects a domain entity (e.g. DC1) and presses the "Update Entity" button. Attributes defined for the domain entity type (e.g. Deck) of the selected entity are

all listed in Figure 6.23 in alphabetical order. The values of these attributes are also listed with some attribute properties, such as attribute type, derivation, role, and group. Whenever the user selects an attribute, the system responds differently according to the attribute type. In numeric attributes, a range of all possible attribute values is listed, so the user has to enter a value within this range. In symbolic attributes, all possible attribute values are listed, so the user has to select one of them. In date attributes, the user has to select a date using the given calendar. In text attributes, the user can enter any value.

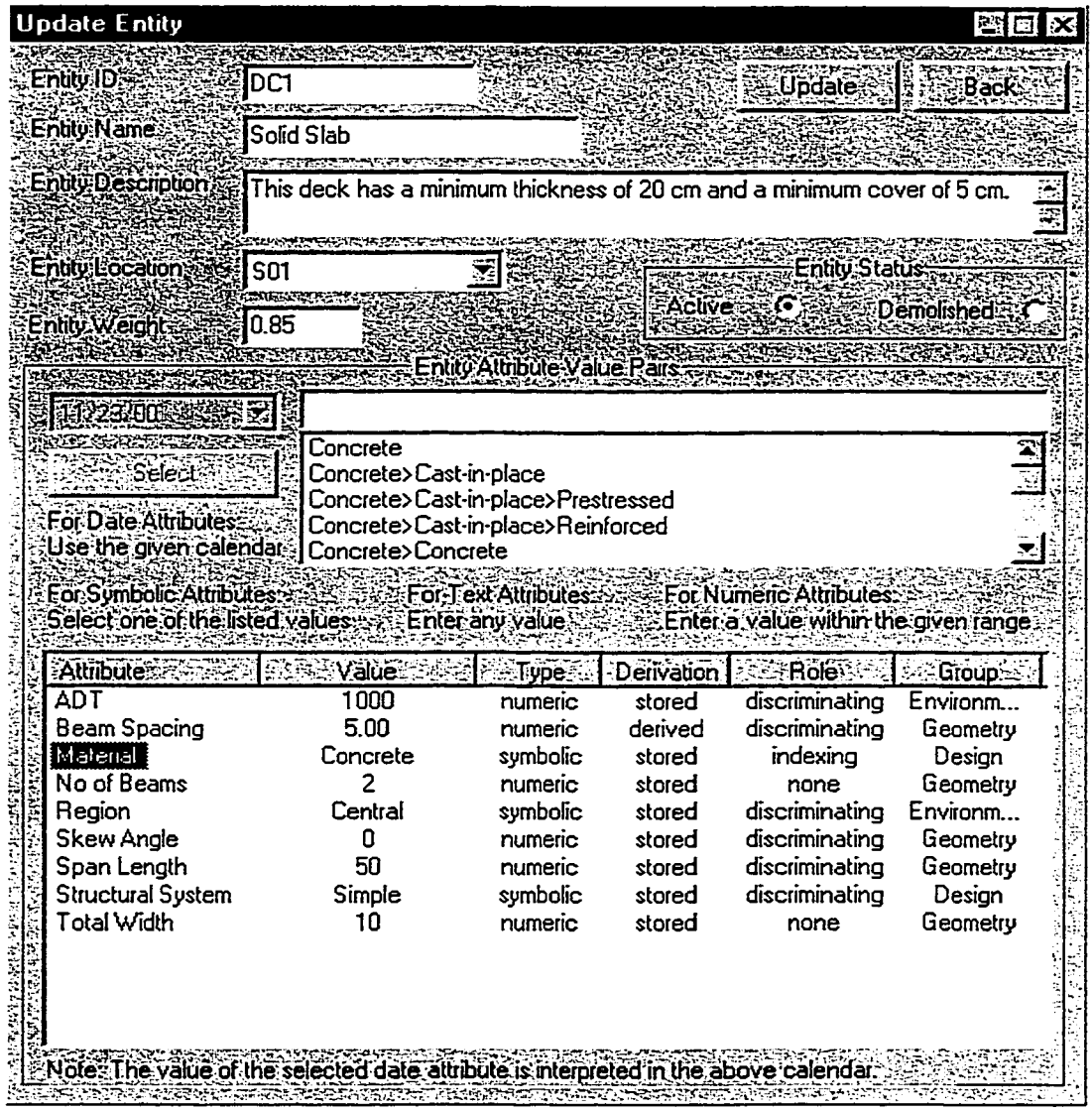


Figure 6.23: Update Entity dialog

The previous dialog is also used to store and update domain entity properties such as entity id, name, description, weight, status, and location. Entity location (e.g. S01) is determined using one of the location identifiers defined for the root entity (e.g. B01). These identifiers are defined in the "Location Identifiers" dialog, shown in Figure 6.24, that appears once the user selects the root entity and presses the "Location Identifiers" button shown in Figure 6.22. In Figure 6.24, different types of location identifiers, such as axes and spans, can be defined. To create a location identifier, the user has to enter or select (if existing) an identifier type, enter a non-null and unique identifier name (e.g. S01), enter an identifier description (if any), and press the "Add identifier" button. Location identifiers that are connected with the selected location identifier are listed in Figure 6.24. These identifiers are selected from a list of all available identifiers and added to/removed from the list of connected identifiers using the "Connect/Disconnect" button. Also, a list of all domain entities located at the selected identifier is provided to simplify finding specific entities in complicated case structures.

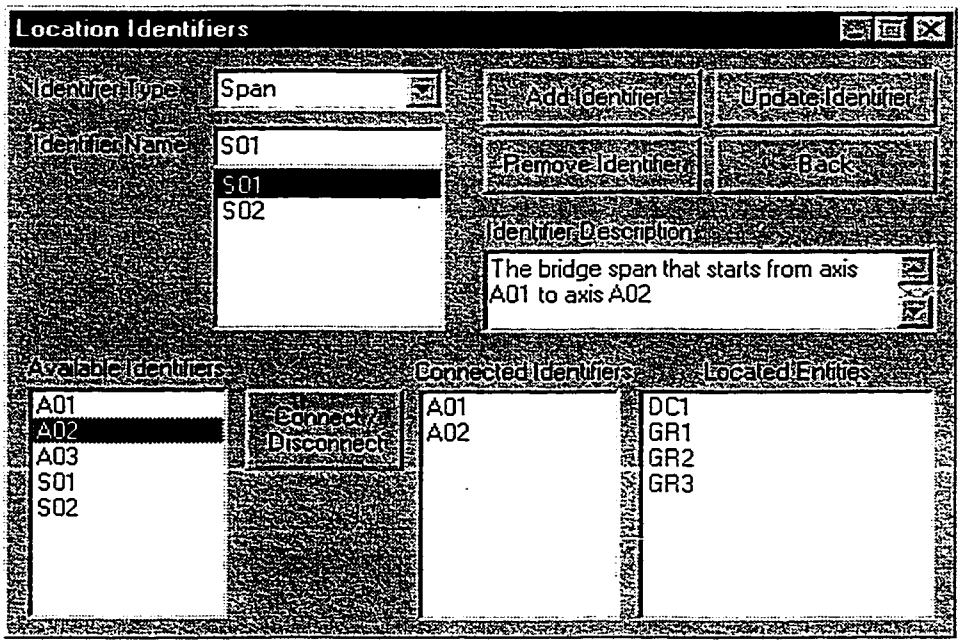


Figure 6.24: Location Identifiers dialog

Storing dynamic data is exactly the same as storing static data except that the user has to create a time-dependent set for every data record that has a different time point. Figure 6.25 shows the "Time-Dependent Sets" dialog that is used to store and update all the dynamic data of a specific domain entity. This dialog appears once the user selects a domain entity and presses the "Time-Dependent Sets" button shown in Figure 6.23. Figure 6.25 shows three time-dependent sets ("I1", "I2", and "I3") that are defined for the "Element Condition" time-dependent set type and belong to the domain entity "DC1". The values stored in "I1" are listed along with their corresponding attributes.

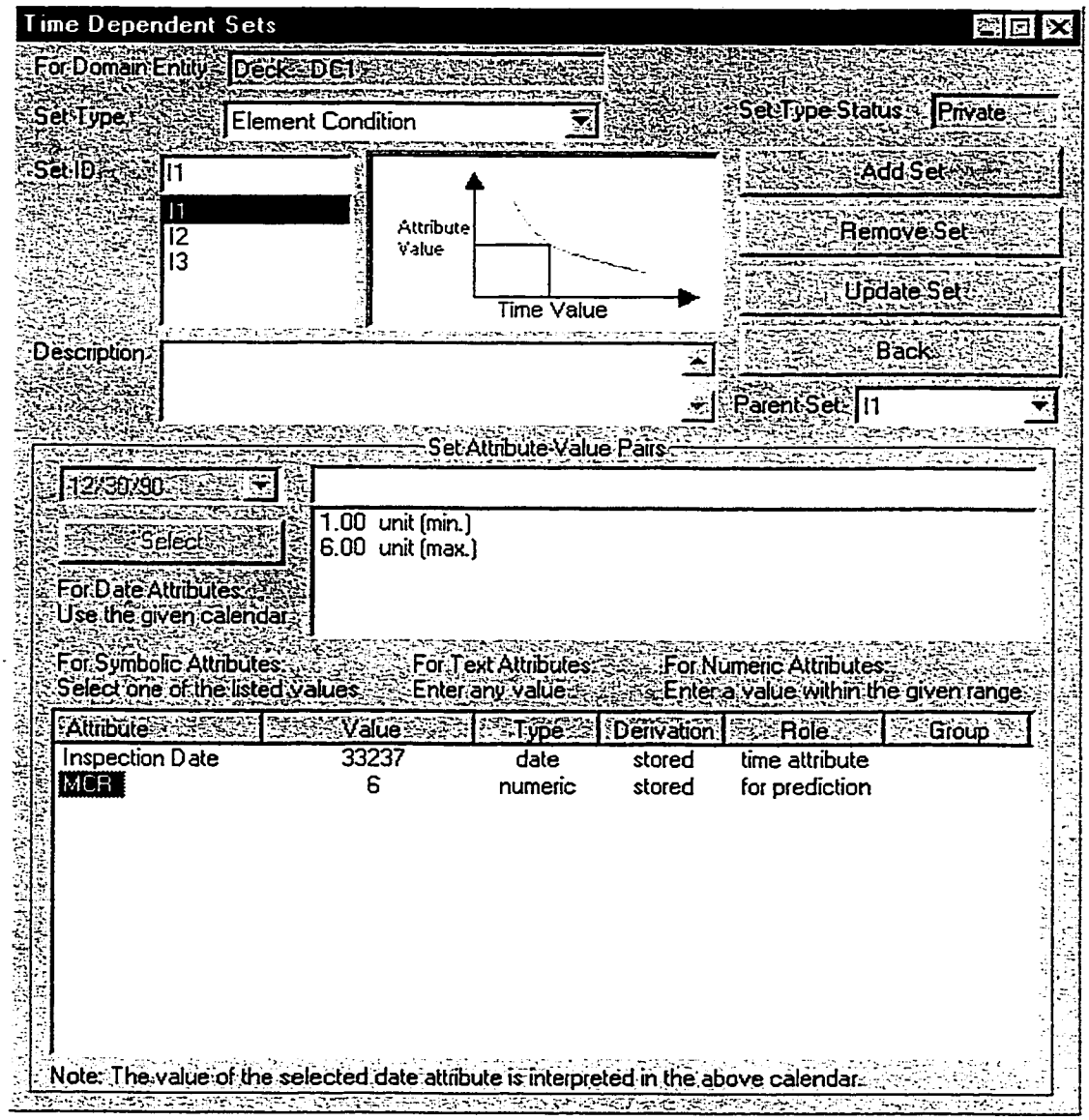


Figure 6.25: Time-Dependent Sets dialog

Entering data from scratch for a large number of domain entities can be a tedious and error-prone task. Hence, the system provides a functionality to transfer data from other data sources into the case library. Figure 6.26 shows the dialog that is used to transfer data from any ODBC data source into domain entities and time-dependent sets defined in the case library. In this dialog, a table called "Deck" from a Microsoft Access 97 database called "Bridge Data" has been selected to be the data source. This table contains inventory data about several bridge decks that are identified using the "Bridge_ID" and "Deck_ID" fields (i.e. primary keys). The values of these fields are used to map deck records from the Access database into deck domain entities. Each deck record corresponds to the deck domain entity that has the same deck id and belongs to a bridge entity that has the same bridge id. The values of the other fields in every record are mapped to the values of the matched attributes in the corresponding domain entity. This dialog can also be used to create domain entities and time-dependent sets from scratch.

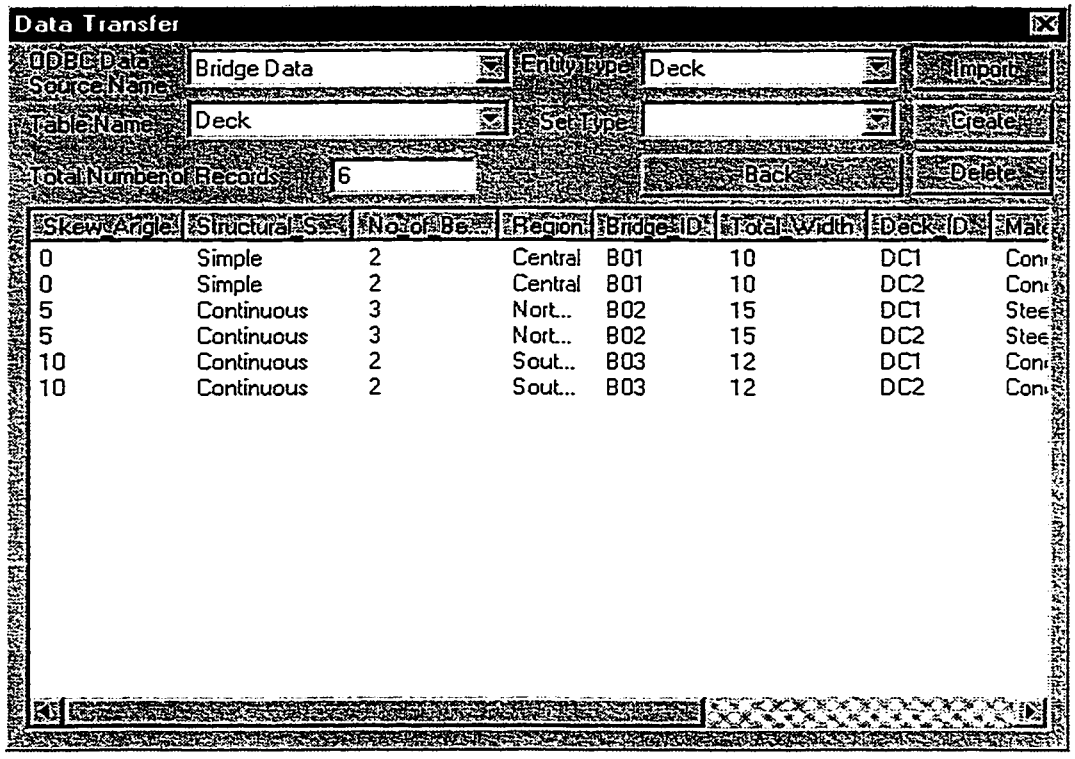


Figure 6.26: Data Transfer dialog

6.3.6 RETRIEVAL AND ADAPTATION MODULE

Whenever the user wants to predict the condition of an infrastructure component, he/she can use the "Case Retrieval and Adaptation" module in CBRMID to search for the case that best matches the query and to adapt its solution. Figure 6.27 shows the "Case Retrieval and Adaptation" dialog that belongs to that module.

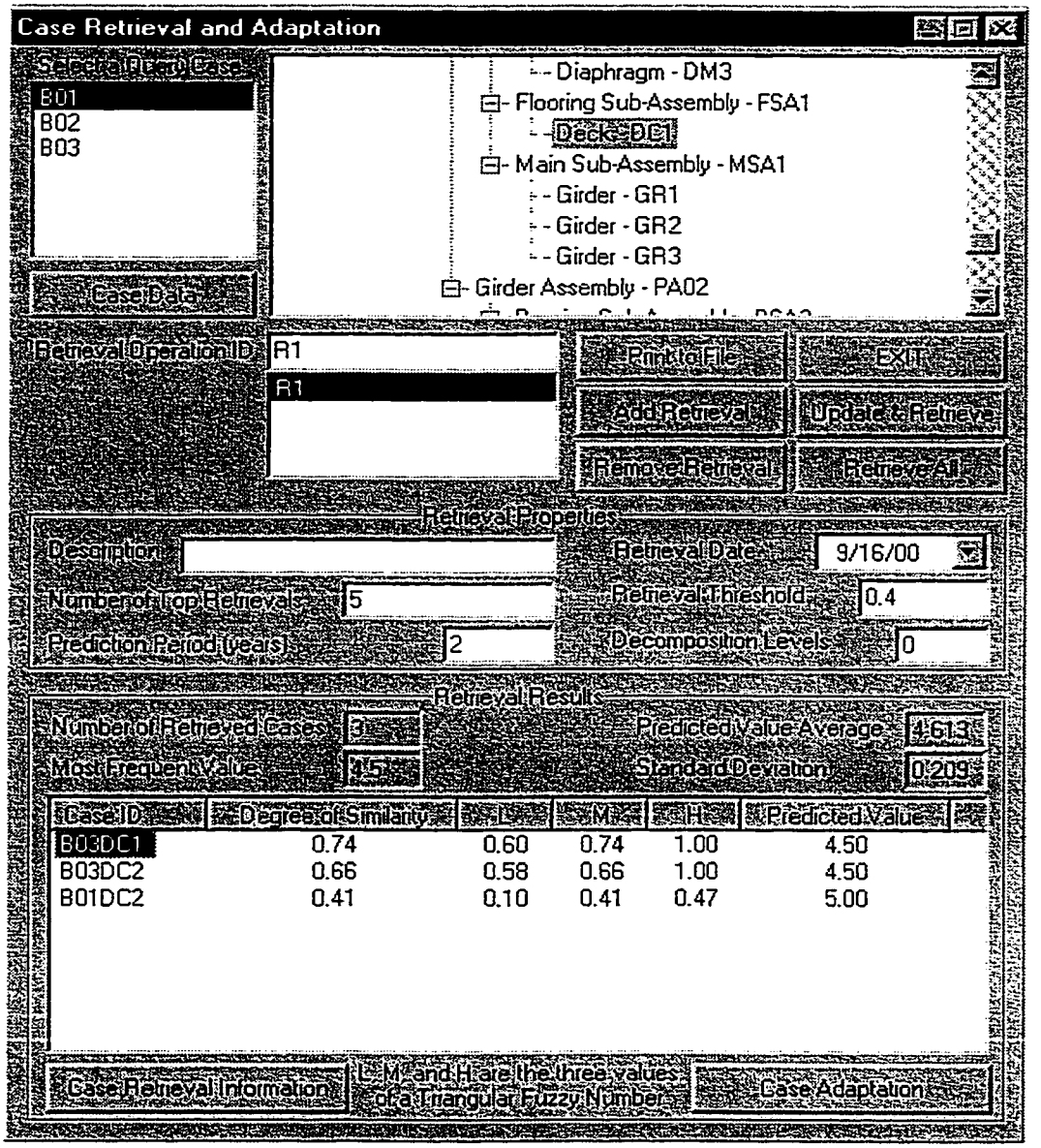


Figure 6.27: Case Retrieval and Adaptation dialog

In this dialog, the user has to select the case (e.g. B01) that contains the query domain entity. The hierarchical decomposition of the selected case will be displayed to allow the user to select the domain entity that represents the query case (e.g. Deck DC1). Detailed information about the query case can be viewed by using the "Case Data" button. Then, the user has to create a retrieval operation that has a non-null and unique identifier (e.g. R1). He/she can also define the properties of this retrieval operation, such as retrieval description, date, threshold, top retrievals, prediction period, and decomposition levels. Retrieval operations created by the user are stored in the database to facilitate carrying out many retrievals using different query cases and various retrieval properties without losing their results. Some of these properties control the retrieval process and affect retrieval results. The number of top retrievals determine the maximum number of best matched cases to be retrieved. The retrieval threshold filters out the retrieved cases with degrees of similarity less than the threshold value. The decomposition levels determine the maximum number of decomposition levels considered in the matching process. The prediction period specifies the time period between the current condition and the predicted condition. In the example shown in Figure 6.27, the top five cases (maximum) were retrieved, a threshold value equal to 0.5 was used, no decomposition levels were considered, and a prediction period equal to two years was conducted.

Retrieval results of the selected retrieval operation (e.g. R1) are presented in three different ways. First, retrieved cases are listed with their identifiers (e.g. B02DC1, B03DC2, and B01DC2), degrees of similarity (e.g. 0.74, 0.66, and 0.42), and predicted values (e.g. 4.5, 4.5, and 5). Each degree of similarity is presented as a crisp number and

as a triangular fuzzy number (i.e. lowest, most likely, and highest values). Second, the average (e.g. 4.613) and standard deviation (e.g 0.209) of the predicted values and the most frequent predicted value (e.g. 4.5) are calculated and displayed. Third, when the user selects any of the retrieved cases, the "Case Retrieval Information" button shown in Figure 6.27 becomes active. This button is used to present the "Case Retrieval Information" dialog shown in Figure 6.28. This dialog shows the details of retrieval calculations according to the retrieval algorithm presented earlier in section 5.4.2.

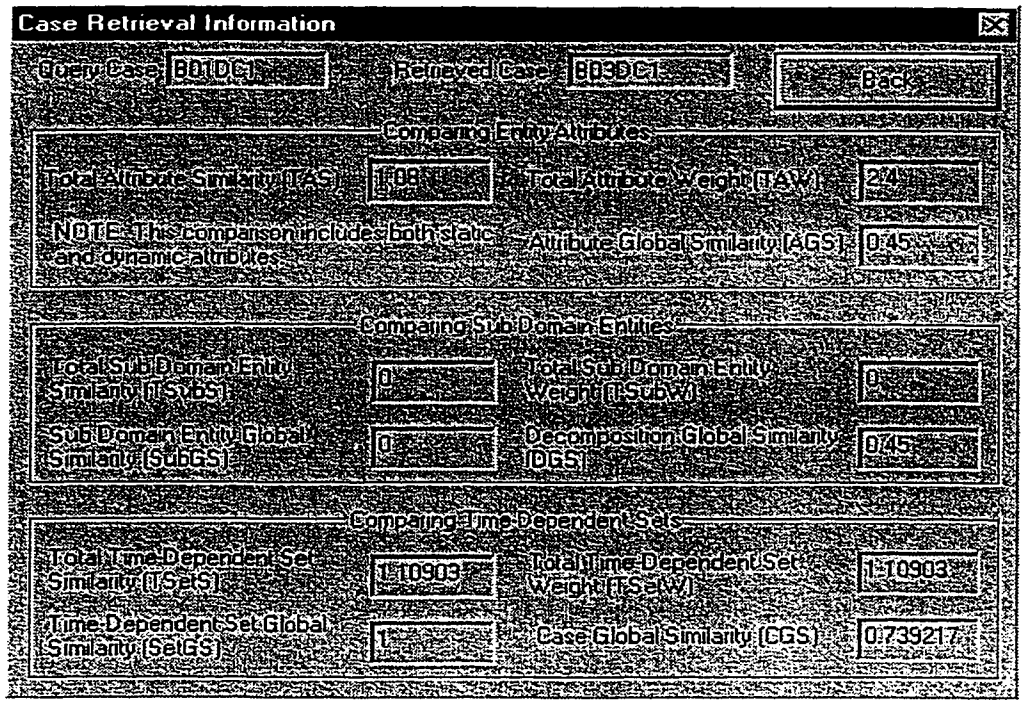


Figure 6.28: Case Retrieval Information dialog

In order to adapt the solution of any of the retrieved cases, the user has to select the case and press the "Case Adaptation" button to present the "Case Adaptation" dialog as shown in Figure 6.29. In this dialog, values of all attributes in both the query and the retrieved cases are listed to clarify the differences between the two cases. These values are ordered as follows: first, values of static attributes; second, the value of the time

attribute corresponding to the attribute used for prediction; third, values of time-dependent attributes different from the one used for prediction; and fourth, values of the attribute used for prediction at the different matched time points. To start the adaptation process, the user has to press the "Run Clips" button to run the Clips program. Then, he/she has to run the batch file called "go.bat" as shown in Figure 6.30 to load all Clips constructs. The Clips inference engine will then apply the adaptation rules to the current facts and record the results in a text file. Once the user presses the "Show Results" button, these results will be displayed in the "Adaptation Results" edit box. According to these results, which may be the adapted solution (as shown in Figure 6.29) or just instructions, the user can update the predicted value shown earlier in Figure 6.27 by pressing the "Update" button. This update revises the calculated average, the standard deviation, and the most frequent value shown in Figure 6.27.

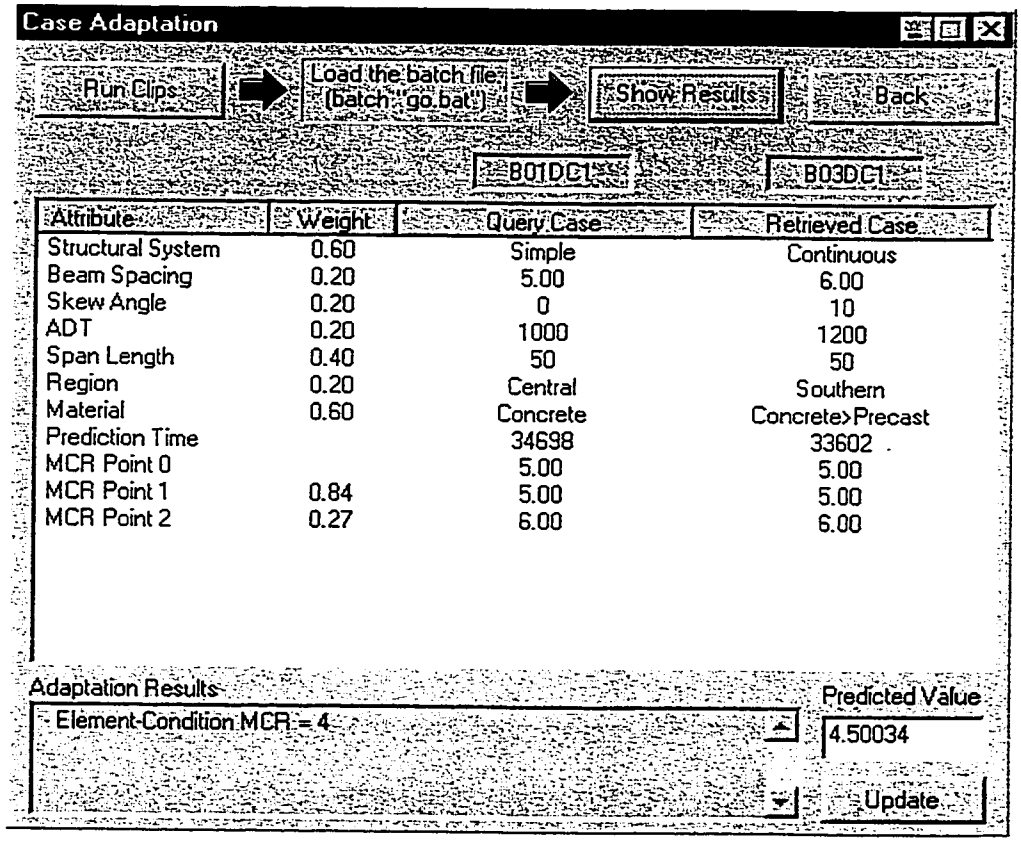


Figure 6.29: Case Adaptation dialog

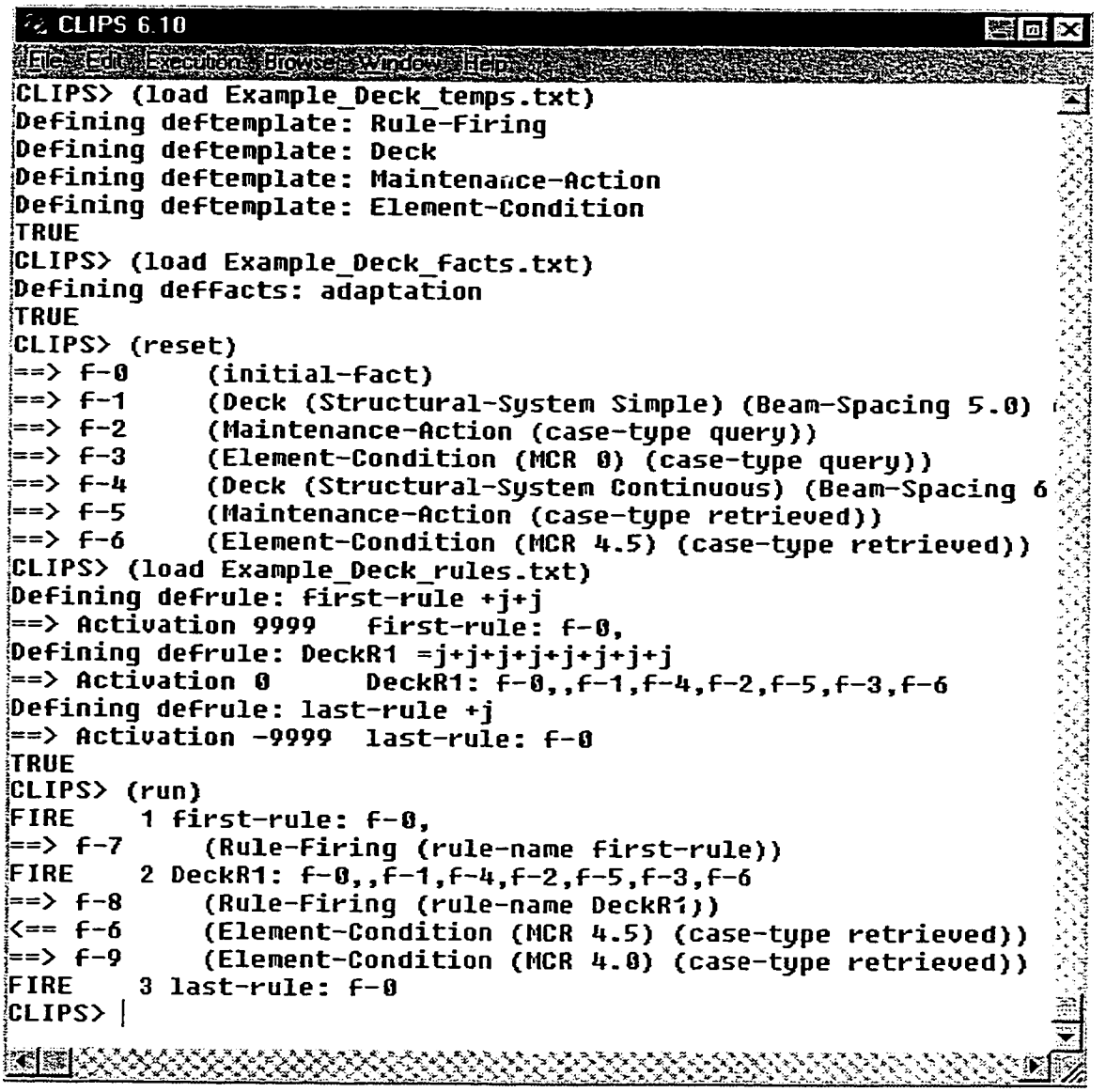


Figure 6.30: Running Clips 6.10 for case adaptation

CHAPTER 7

SYSTEM TESTING AND EVALUATION

7.1 INTRODUCTION

System testing is the last sub-phase of the build phase in the development cycle of any software system (refer to section 3.3). The word "testing" does not mean debugging the code, which has already been done during the code generation process in the system implementation sub-phase, but it means measuring the system performance. The system testing consists mainly of two processes (Ng and Smith 1998): system verification and system validation.

The verification process is directed to check the consistency and completeness of the software system. The system is considered inconsistent if it produces contradicting outputs whenever similar inputs are used (Roddis and Bocox 1997). The system is considered incomplete if it cannot respond to all possible situations that arise within the domain (Botten el al. 1989). In the CBR system developed for modeling bridge deterioration, cases that constitute the case library are those bridge records stored in the database of the host BMS. This means, all bridge records stored in the BMS database have to be included in the case library as cases regardless of their consistency and completeness. No user intervention is allowed to remove any contradicting cases or to add new cases from other data sources in order to maintain the consistency and completeness of the system. Therefore, system verification will not be an effective

process for the developed CBR system but it will be carried out for the purpose of data analysis only.

The purpose of the validation process is to check the accuracy, sensitivity, and efficiency of the software system. System accuracy determines how close the solutions provided by the system are to reality (Ng and Smith 1998). System sensitivity determines how much these solutions fluctuate when changing the system parameters. System efficiency determines how competitive the system is with other systems in terms of running time and capacity (Roddis and Bocox 1997). These three checks are crucial in the CBR system developed for modeling bridge deterioration because they determine whether or not the CBR approach is appropriate for solving such a problem.

In this chapter, data collected about Quebec bridges were used to develop a "proof of concept" CBR application for modeling bridge deck deterioration using the developed CBR shell CBRMID and to test its performance with actual cases. In the "Testing Data" section, a detailed description of these data is presented. The "Data Analysis" section reports the results of the statistical analysis carried out to check the quality of the data collected and to obtain knowledge from it. The "System Validation" section discusses the procedures followed in validating the developed system along with the results of this validation. The last section presents an overall evaluation of the developed system based on the testing results. It should be noted that the word "system" mentioned throughout this chapter means the CBR approach, the developed CBR shell (CBRMID), and the "proof of concept" application for modeling bridge deck deterioration.

7.2 TESTING DATA

The data used in testing the system developed in the present investigation were obtained from the Ministry of Transportation of Quebec (MTQ) database. This database is part of a comprehensive system for managing different highway structures such as bridges, culverts, tunnels, retaining walls, etc. The architecture of this comprehensive system is presented in Appendix E. The MTQ database consists of 9500 province-owned highway structures that are categorized into eight categories of structures: culverts, slab bridges, beam bridges, box-girder bridges, truss bridges, arch bridges, cabled bridges, and other structures. Figures of these structure types and the number of structures in each category are also presented in Appendix E. The MTQ database includes three types of data for each of the 9500 highway structures (refer to bridge data categories presented earlier in section 4.2):

i. **Inventory data** consist of approximately 220 data items that can be categorized into administrative, technical, and descriptive data. For more information about these data items, refer to MTQ (1997).

ii. **Inspection data** consist of detailed visual inspection of 135 bridge elements (maximum) collected every 3 years (maximum) since 1993. These data are collected using 21 different inspection forms, each of which corresponds to a group of correlated bridge elements, such as foundation elements, truss elements, deck elements, etc. A list of these inspection forms along with an example of the bridge deck inspection form are presented in Appendix E. Each inspection form includes the material and performance condition ratings of the inspected elements in addition to some remarks and recommendations regarding the maintenance of these elements

(MTQ 1995). In order to provide the overall condition of a group of elements, conditions of individual elements are aggregated using balance factors (i.e. entity weights) that represent the importance of each element with respect to the material and performance condition relative to the other elements in its inspection form. Balance factors of all bridge elements are listed in Appendix E.

iii. **Maintenance data** represent the major maintenance actions that are expected to be made in the future. The cost and the recommended period of each action are roughly estimated. Maintenance data of the actions that were taken in the past are collected by the regional offices and not recorded in the MTQ database. Although the developed system is designed to consider the effect of previous maintenance actions on the predicted condition, the data of these actions will not be included in testing the system because of its unavailability.

The CBR application developed in this chapter focuses on modeling the deterioration of only one bridge component to prove the concept. The concrete bridge deck was selected for that purpose because it is the bridge component that has the highest deterioration rate due to its direct and continuous exposures to traffic, weather, and deicing chemicals (Freyermuth et al. 1970; Busa 1985). This selection was also based on the fact that data about concrete bridge decks are more readily available than any other bridge component (in the MTQ database and in the literature on bridge deterioration). The selection of concrete bridge decks is sufficient to prove the concept since the same procedures can be followed with any other bridge component and construction material.

The premature deterioration of concrete bridge decks has motivated bridge engineers and researchers to study the factors that affect this deterioration since the early 1970's. These factors are categorized and listed in Table 7.1 according to the two studies prepared by Carrier and Cady (1973) and Busa (1985). Although some of these factors are highly important for modeling deck deterioration, such as concrete cover, protective system, construction practice, use of de-icing chemicals, and deck maintenance actions, these factors were not considered in the developed application because the data relevant to these factors are not recorded or even collected by the MTQ.

Category	Factor
1- Material	Water-cement ratio Air content Flexural strength Aggregate source and size Admixtures
2- Design	Superstructure type Structural system Beam spacing Span length Deck width Slab thickness Concrete cover Deck grade Skew angle Deck stiffness (parapets) Protective system (wearing surface)
3- Construction	Mixing method Construction practice Curing method Finishing technique Temperature at construction
4- Environment (Natural Sources) (Man-made Sources)	Moisture (humidity and precipitation) Freeze-thaw cycles Average daily traffic Percentage of truck traffic Use of de-icing chemicals Linseed oil application Deck maintenance actions

Table 7.1: Factors affecting bridge deck deterioration (Carrier and Cady 1973; Busa 1985)

A review of the currently available bridge deck deterioration models was carried out to determine the factors considered by these models. Table 7.2 summarizes the findings of this review by listing the factors recognized by four deterioration models used in advanced State DOT's in the U.S. These factors are: highway class, region, average daily traffic (ADT), average daily truck traffic (ADTT), material, structural system, age, wearing surface, and skew angle. This table shows that none of these models was able to even consider the factors listed in Table 7.2. The next sections demonstrate the capability of the developed system to consider all these factors in addition to some other factors that account for deck past conditions, girder condition, girder spacing, and girder material.

State	Model Type	Highway Class	Region	ADT	ADTT	Material	Structural System	Age	Wearing Surface	Skew Angle
Virginia (Scherer and Glagola 1994)	Markovian	✓	✓	✓	✓	✓	✓	✓		
Nevada (Sanders and Zhang 1994)	Regression		✓	✓	✓	✓		✓		
North Carolina (Abed-Al-Rahman and Johnston 1995)	Curve-Fitting	✓	✓	✓		✓		✓		
Indiana (Madanat et al. 1997)	Markovian	✓	✓	✓		✓	✓	✓	✓	✓

Table 7.2: Factors considered in modeling bridge deck deterioration by four States DOT's in the U.S.

Concrete bridge decks in beam bridges of types 41 (reinforced cast-in-place concrete beams), 42 (prestressed precast concrete beams), and 43 (prestressed cast-in-place concrete beams) were selected to construct the case library of the developed system (figures of these type are shown in Appendix E). This is because these types of beam bridges are the dominant in Quebec; the total number of such bridges is 2382, which

constitutes 25% of Quebec bridges. Furthermore, these bridges provide the majority of concrete decks (cases), which increases the probability of finding matching cases in the case library. Figure 7.1 shows a photo of a bridge of type 41 that passes over Highway 20 between Montreal and Quebec City.

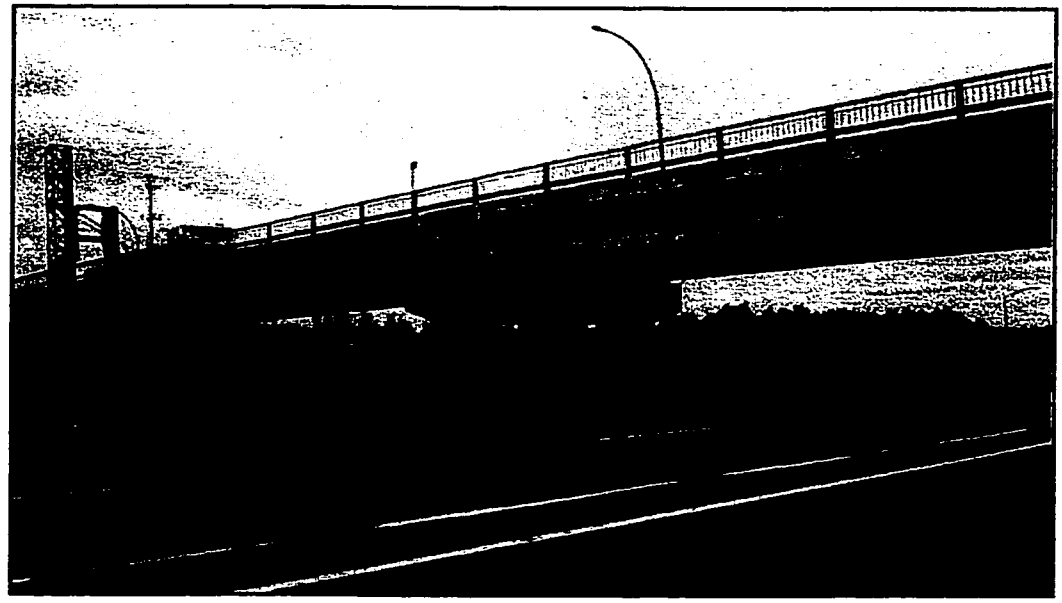


Figure 7.1: Example of a beam bridge of type 41

Since the BMS in Quebec has been put in operation recently, some bridge records have incomplete inventory data, one inspection record only, or missing condition data. All these records were eliminated from the testing set to avoid misleading testing results. Due to the unavailability of bridge maintenance data, bridge records that have positive deterioration rates are also eliminated. These positive rates indicate that major maintenance actions have been performed during the observation period. These actions affect the predicted condition greatly and cannot be incorporated unless their maintenance data become available. Table 7.3 summarizes some information about the set of bridges used in the system testing.

No. of Inspections	2 Inspections	3 Inspections	Total
No. of bridges	107	133	*240*
No. of Inspection Records	214	399	*613*
No. of Bridge Spans	262	259	*521*
No. of Span Conditions	524	777	*1301*

Table 7.3: Number of bridge inspections and bridge spans in the testing set

In order to represent bridge deck cases used in constructing the case library, a domain model that consists of five domain concepts obtained from the conceptual model of bridges presented earlier in section 4.4.3 is used. These five concepts consist of three domain entity types (Bridge, Deck, and Girder) and two time-dependent set types (Bridge Inspection and Element Condition). This domain model is a subset of the domain model presented earlier in Chapter 6. Attributes defined for these concepts and used to describe the contents of bridge deck cases are listed in Table 7.4. All possible attribute values, as defined by MTQ, are presented as ranges for numeric attributes and as enumerated values in Tables 7.5 to 7.10 for symbolic attributes. Attribute values that exist in the testing set are also listed in the column titled "Used Values". Among these 17 attributes, four attributes (total width, no. of beams, construction date, and inspection date) do not participate in the reasoning process; they are used in deriving the values of two other attributes (beam spacing and age) that do participate in the reasoning process. Although MCR is the attribute used for prediction, its past values are considered as a deterioration factor that affects the predicted concrete deck condition. This attribute is also reused by the "Girder" domain entity type to represent the condition of girders interacting with a bridge deck. Although the "Material" attribute is reused by both "Deck" and "Girder" domain entity types, its values for the "Deck" domain entity type were not considered as a deterioration factor because the testing is limited to concrete bridge decks only.

No.	Domain Entity Type or Time-Dependent Set Type	Attribute	Possible Values	Used Values	Unit	Derivation
1	Bridge	Region	Table 7.5	Table 7.5	N.A.	Stored
2		Highway Class	Table 7.6	Table 7.6	N.A.	Stored
3		ADT	0 - 999999	0 - 136000	pcu	Stored
4		Truck%	0 - 100	0 - 38	N.A.	Stored
5	Bridge Inspection	Inspection Date	1993 - 1999	1993 - 1999	year	Stored
6	Deck	Span Length	0 - 1000	5.8 - 39.70	meter	Stored
7		Skew Angle	0 - 65	0 - 54.94	degree	Stored
8		Material	Table 7.7	Table 7.7	N.A.	Stored
9		Total Width	0 - 120.9	4.7 - 33.36	meter	Stored
10		Beams Spacing	0 - 61	1.07 - 5.55	meter	Derived
11		Structural System	Table 7.8	Table 7.8	N.A.	Stored
12		Wearing Surface	Table 7.9	Table 7.9	N.A.	Stored
13		Construction Date	1800 - 1999	1923 - 1996	year	Stored
14		No. of Beams	0 - 99	2 - 16	each	Stored
15	Girder	Material	Table 7.10	Table 7.10	N.A.	Stored
16	Element Condition	MCR	1 - 6	1 - 6	rating	Stored
17		Age	0 - 200	1 - 76	year	Derived

N.A. means Not Applicable

Table 7.4: List of attributes used to represent case contents

Region	Code in Figure 7.2	Possible Values	Used Values
31	2	Gaspésie	Eastern
33	3	Bas-Saint-Laurent	
32	4	Québec Greater Area	
34	6	Chaudière-Appalaches	
38	7	Mauricie	Central
53	8	Eastern Townships	
54	9	Montérégie	
55	10	Lanaudière	
55	11	Laurentides	
52	12	Montréal	
51	19	Laval	
57	20	Centre-du-Québec	
56	13	Outaouais	Western
58	14	Abitibi-Témiscamingue	
36	15	Saguenay–Lac-Saint-Jean	Northern
39	16	Manicouagan	
35	17	Duplessis	
37	18	Far North	

Table 7.5: Values of region attribute

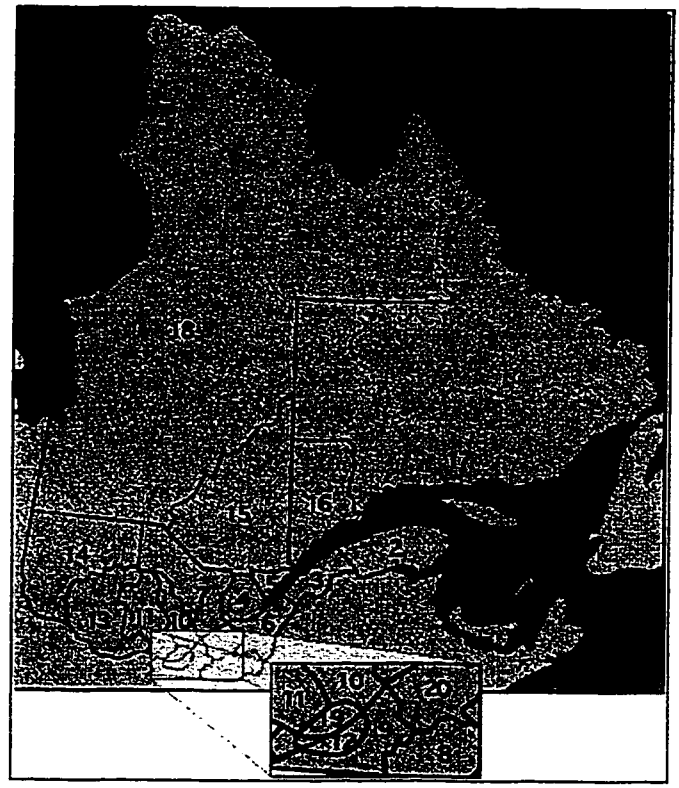


Figure 7.2: Regions of Quebec province

Highway Class	Possible Values	Used Values
10	Express way	Express
15	Express way (political)	
20	National	National
25	National (political)	
30	Regional	Regional
40	Collector	Collector
51	Local 1	Local
52	Local 2	
53	Local 3	
60	Resources access	N.A.
91	Foot path	
92	Railway	
93	Others	

Table 7.6: Values of highway class attribute

Deck Material	Possible Values	Used Values
1	Reinforced Concrete	Reinforced Concrete
2	Timber	N.A.
3	Orthotropic	
4	Grating	
5	Concrete with Corrugated Sheets	
6	Concrete with Transversal Prestressing	
7	Concrete with Alternating Trans. Prest.	
8	Prestressed Wood	
9	Other	

Table 7.7: Values of deck material attribute

Structural System	Possible Values	Used Values
S	Simple	Simple
C	Continuous	Continuous

Table 7.8: Values of structural system attribute

Wearing Surface	Possible Values	Used Values
1	Bituminous Mix	Bituminous Mix
2	Cement Mix	Cement Mix
3	Wood	N.A.
4	Rubber	
5	Steel Grid	
6	Gravel	
7	Other	

Table 7.9: Values of wearing surface attribute

Girder Material	Possible Values	Used Values
1	Cast-in-Place Reinforced Concrete	Cast-in-Place Reinforced Concrete
2	Cast-in-Place Prestressed	Cast-in-Place Prestressed
3	Precast Prestressed	Precast Prestressed
4	Steel	N.A.
5	Wood	
6	Others	

Table 7.10: Values of girder material attribute

7.3 DATA ANALYSIS

A statistical analysis is carried out in order to explore the quality (i.e. the consistency and completeness) of the data set selected for the system testing and to obtain retrieval knowledge (i.e. weights and degrees of similarity) from these data. In this analysis, associations between the different attributes listed earlier in Table 7.4 and the attribute used for prediction are measured. These associations help in determining whether the data set is inconsistent or missing some important attributes. Furthermore, these associations can provide some knowledge regarding the relative importance of attributes and the degrees of similarity among their values.

In general, there are two different statistical methods that can be used to assess the association between two variables according to their types (Moore and McCabe 1993): measuring the coefficient of correlation and performing the analysis of variance (ANOVA) test. The coefficient of correlation measures the strength of the linear association between two quantitative variables. This coefficient takes a value between –1 and +1. It is negative if one variable tends to increase as the other variable decreases, and positive if the two variable tend to increase or decrease together. The analysis of variance (ANOVA) test compares several population means and consequently measures the association between one quantitative variable and another qualitative variable that is used to split the quantitative variable into several populations. This test has a null hypothesis "Ho" that the population means are all equal and an alternative hypothesis "Ha" that at least one population mean is different.

In order to apply any of these two statistical methods, the response variable (i.e. the dependent variable) has to be distinguished from the explanatory variables. In this data analysis, the deck MCR was not selected to be the response variable because the value of this variable is greatly affected by the maintenance actions taken in the past, which are not recorded. Therefore, the rate of deck deterioration between every two successive deck inspections was selected instead. The value of this variable is more accurate as long as no maintenance actions have been taken within the inspection period. In order to have positive values of the response variable, which lead to easier interpretation of results, the rate of deterioration is calculated as follows:

$$R_i = \frac{MCR_i - MCR_{i+1}}{t_{i+1} - t_i}$$

Where,

R_i	Rate of deterioration at inspection i
MCR_i	Material condition rating at inspection i
t_i	Time at inspection i

The MCR used in the previous equation is a continuous float number that represents the overall deck condition as the aggregation of the MCR of five different portions of deck: two exterior faces, two end portions (each has a length equal to two times the girder depth), and the middle portion (MTQ 1995). These MCR's are aggregated using the balance factors listed in Appendix E for the inspection sheet "E". The calculated rate of deterioration is also a continuous float number that represents the decline in the overall slab condition in one year. This explanatory variable differs from the one reported by Madanat and Ibrahim (1995) as discrete integer number representing the number of drops

in the deck condition during one inspection period. The total number of records used in this analysis is 780. This number is obtained from Table 7.3 and based on the fact that bridge spans with two inspections resulted in one record of deterioration rate and bridge spans with three inspections resulted in two records of deterioration rate. This statistical analysis was carried out using the Minitab Statistical Software Release 12.

Figure 7.3 shows the correlation matrix (only the lower triangle is shown because of symmetry) that displays the calculated coefficient of correlation (shown in the top of each cell) between each pair of quantitative attributes. This correlation matrix also displays p-values (shown below the coefficients of correlation) for the hypothesis test of the coefficient of correlation being zero. This hypothesis test is a two-tailed test that has a null hypothesis (Ho: $r = 0$) and an alternative hypothesis (Ha: $r \neq 0$), where "r" is the coefficient of correlation. Using a level of significance $\alpha = 0.05$, the common practice in hypothesis tests (Sincich 1994), all coefficients of correlation that have p-value greater than or equal to 0.025 are considered insignificant.

	ADT	Truck Percent	Span Length	Skew Angle	Beam Spacing	Age	Deck MCR	Girder MCR
Truck Percent	-0.11 0.00							
Span Length	0.21 0.00	-0.01 0.76						
Skew Angle	-0.06 0.07	0.21 0.00	-0.01 0.76					
Beam Spacing	0.14 0.00	-0.05 0.16	-0.01 0.75	-0.06 0.08				
Age	-0.25 0.00	-0.16 0.00	-0.45 0.00	-0.07 0.06	-0.02 0.62			
Deck MCR	0.12 0.00	0.03 0.40	0.34 0.00	-0.18 0.00	-0.02 0.65	-0.39 0.00		
Girder MCR	0.08 0.03	0.10 0.01	0.16 0.00	-0.06 0.08	-0.01 0.77	-0.40 0.00	0.54 0.00	
Deterioration Rate	-0.17 0.00	0.04 0.25	-0.09 0.02	0.02 0.50	0.04 0.23	0.11 0.00	0.02 0.55	-0.03 0.37

Figure 7.3: Correlation matrix

The last row in the above correlation matrix contains the associations of all quantitative variables with the response variable "Deterioration Rate". In this row, only the associations with the "ADT", "Span Length", and "Age" are significant. The negative association between the "Deterioration Rate" and the "ADT" is natural and logical because the MTQ gives great attention to the traffic volume on highway structures in ranking them for eligibility for maintenance actions. This means, structures with higher traffic volume and on essential highways are subjected to preventive maintenance more often than other structures with less traffic volume or on local highways. The negative association between the "Deterioration Rate" and the "Span Length" is contrary to what was reported in the literature. This is probably due to the fact that long span bridges are usually made of prestressed concrete girders, which lead to less deteriorating decks, while short span bridge are made of reinforced concrete girders, which lead to more deteriorating decks. This fact was proven in Table 7.11 that shows the average span length in each category of girder material. The positive association between the "Deterioration Rate" and the "Age" means that older bridge decks have higher deterioration rates, which coincides with the literature on bridge deterioration (Bulusu and Sinha 1997). It should be mentioned that the highest coefficient of correlation in the above matrix is between the "Deck MCR" and "Girder MCR" variables, which confirms the importance of considering the interaction among different bridge components.

Girder Material	No. of Bridges	Average Span Length (in meters)	Standard Deviation
Cast-in-Place Reinforced	555	18.888	6.249
Cast-in-Place Prestressed	55	26.402	6.205
Precast Prestressed	170	28.332	7.476

Table 7.11: Average span length corresponding to each girder material

Figure 7.4 shows the results of five one-way ANOVA tests that were carried out to measure the association between the response variable and five qualitative variables: "Highway Class", "Region", "Structural System", "Wearing Surface", and "Material" respectively. In each test, two tables are provided: the first table, on the left, shows the significance of the ANOVA test while the second table, on the right, shows some descriptive statistics about each value (level) in the qualitative explanatory variable.

Source	DF	SS	MS	F	P
Highway Class	4	0.52	0.1311	7.66	0
Error	775	13.27	0.0171		
Total	779	13.79			

Level	N	Mean	StDev
Collector	82	0.102	0.124
Express	145	0.032	0.060
Local	167	0.095	0.131
National	258	0.087	0.134
Regional	128	0.109	0.179

Source	DF	SS	MS	F	P
Region	3	0.56	0.1859	10.9	0
Error	776	13.24	0.0171		
Total	779	13.79			

Level	N	Mean	StDev
Central	227	0.099	0.1384
Eastern	255	0.106	0.1432
Northern	168	0.070	0.1223
Western	130	0.032	0.0964

Source	DF	SS	MS	F	P
Structure System	1	0.01	0.0121	0.68	0.408
Error	778	13.78	0.0177		
Total	779	13.79			

Level	N	Mean	StDev
Continuous	216	0.090	0.154
Simple	564	0.081	0.124

Source	DF	SS	MS	F	P
Wearing Surface	1	0.04	0.038	2.15	0.143
Error	778	13.76	0.0177		
Total	779	13.79			

Level	N	Mean	StDev
Bituminous Mix	742	0.085	0.135
Cement Mix	38	0.053	0.082

Source	DF	SS	MS	F	P
Material	2	0.09	0.0431	2.44	0.088
Error	777	13.71	0.0176		
Total	779	13.79			

Level	N	Mean	StDev
Cast-in-Place Prestressed	55	0.064	0.124
Cast-in-Place Reinforced	555	0.090	0.140
Precast Prestressed	170	0.069	0.111

Figure 7.4: ANOVA tests for five qualitative variables

In each table on the left side, the sources of each variation "SS" in the values of the "Deterioration Rate" variable, the value of that variation, and the corresponding degree of freedom "DF" are listed. The *F* test statistic is calculated by dividing the mean of squares

"MS" (where, MS = SS / DF) of the qualitative variable over the "MS" of errors. This test compares the variation resulting from the qualitative variable to the variation resulting from errors. This test shows that the "Highway Class" and "Region" are the only significant variables (using a level of significance $\alpha = 0.05$) among the five explanatory variables. By looking at the tables on the right side, it can be concluded that bridges on express highways or bridges in the western regions of Quebec have lower deterioration rates. This is may be because bridges on express highways are like those with high traffic volume: they have priority for preventive maintenance actions due to their importance for public use. On the other hand, bridges on the western regions of Quebec are not close to any coasts and, consequently, they are not subjected to the high moisture and salt of the coastal regions (Carrier and Cady 1973). It should be noted that Quebec regions presented in Table 7.5 are categorized into northern, central, eastern, and western regions based on their geographical location only and regardless of the climatic characteristics of each region, which are represented by the environmental factors listed earlier in Table 7.1. This is because data about these factors are not available.

In order to check the consistency and completeness of the data set for predicting the bridge deck deterioration rate, a multiple linear regression model was built. This model was designed to consider the effect of all quantitative and qualitative explanatory variables simultaneously on the response variable. Every qualitative variable that has "k" categories was replaced by "k-1" dummy variables, each of which takes a value equal to either 0 or 1 and represents one category. Having zero values for all "k-1" categories means that the absent category has a value equal to 1. Table 7.12 lists all the predictors

(the constant and explanatory variables) of the regression model along with their coefficients, standard deviations, t-ratios, and p-values. The t-ratio and p-value given for each variable belong to the two-tailed test of hypothesis for the existence of that variable. The value of the standard error of estimate "S", and coefficients of determination "R-Sq" , "R-Sq(adj)" are also provided.

Predictor	Coefficient	Standard Deviation	T-ratio	p-value
Constant	0.0019600	0.0596700	0.03	0.97
ADT	0.0000001	0.0000004	0.15	0.88
Truck Percent	0.0011660	0.0010150	1.15	0.25
Span Length	-0.0014837	0.0007677	-1.93	0.05
Skew Angle	-0.0000008	0.0003176	0.00	1.00
Beam Spacing	0.0201190	0.0099210	2.03	0.04
Deck MCR	0.0262010	0.0083550	3.14	0.00
Girder MCR	-0.0029200	0.0061490	-0.47	0.64
Age	0.0004568	0.0004425	1.03	0.30
Express Highway	-0.0767200	0.0208300	-3.68	0.00
National Highway	-0.0207300	0.0147400	-1.41	0.16
Regional Highway	-0.0121900	0.0168900	-0.72	0.47
Collector Highway	-0.0039100	0.0181900	-0.22	0.83
Simple Structure System	-0.0438000	0.0123500	-3.55	0.00
Northern Region	-0.0380200	0.0170200	-2.23	0.03
Eastern Region	-0.0028200	0.0127000	-0.22	0.82
Western Region	-0.0855200	0.0162100	-5.27	0.00
Cement Wearing Surface	-0.0659600	0.0234800	-2.81	0.01
Precast Material	0.0238600	0.0163800	1.46	0.15
Prestressed Material	-0.0568700	0.0319600	-1.78	0.08

S = 0.1267 R-Sq = 11.50% R-Sq (adj) = 9.30%

Analysis of Variance

Source	DF	SS	MS	F	P
Regression	19	1.59054	0.08371	5.21	0
Residual	760	12.20286	0.01606		
Total	779	13.7934			

Table 7.12: Results of the multiple linear regression model

Although the explanatory variables presented in Table 7.12 represent significant deterioration factors, the results of the linear regression model show that most of these variables are insignificant (p-value > 0.025). This insignificance along with the low coefficient of determination of the regression model are indicators that the available cases are inadequate to provide a complete coverage for the different situations. They are also indicators that there are some other variables that are unobserved and/or unrecorded, which result in inconsistency problems. It should also be noted that the linear regression model is an approximation used for simplicity because of the large number of variables, but using non-linear regression models may result in better estimates.

7.4 SYSTEM VALIDATION

The literature on CBR systems includes three different methods for system validation. In Roddis and Bocox (1997), a case-based system for detecting steel bridge fabrication errors (CB-BFX) was validated by comparing its solution with the solutions provided by a knowledge-based expert system that has been in use for many years. In Ng and Smith (1998), a case-based system for evaluating contractor prequalification (EQUAL) was validated by comparing its solutions with the solutions provided by domain experts and semi-experts (e.g. a Turing test). In Ardit and Tokdemir (1999), a case-based system for predicting the outcome of construction litigation was validated by comparing its solutions with the solutions of real-world cases that were not stored in the case library. The last method of validation is widely used in testing AI systems, such as ES, ANN, and CBR systems, because it does not require any additional resources, either systems or humans. This method of validation is selected to evaluate the accuracy of the solutions provided

by the developed system because it makes its comparisons with actual cases and not with human judgement, which is subjective, or with other systems, which may be erroneous. The procedures followed in this validation are described next.

Among the 240 bridges used in the data analysis, 133 bridges were used in the system validation because they have three inspection records. These 133 bridges have a total of 259 spans, each of which represents a case. The last inspection record in about 30 cases was eliminated. This record contains the deck condition that will be used in validating the accuracy of the solution provided by the system. To obtain fair and unbiased results, two different groups of cases, each consisting of 30 cases, were selected at random. Group 1 is used to validate the system and refine its parameters (i.e. training group), while Group 2 is used to confirm this validation (i.e. testing group).

For each group, two different sets of attribute weights were used. In the first set, the same numeric value was assigned to all attribute weights assuming that all attributes have the same importance. In the second set, attribute weights were adjusted manually according to their relative importance based on knowledge obtained from the literature and from the data analysis carried out in the previous section. These weights were refined in several iterations to ensure that the system accuracy was improving significantly. In each iteration, attribute weights were changed to provide better testing results with the training group (Group 1), while the testing group (Group 2) was used to check the effectiveness of these changes.

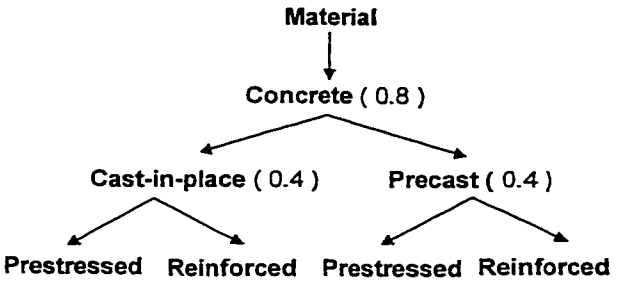
Degrees of similarity among attribute values were also estimated based on knowledge obtained from the literature and the data analysis. The same degrees of similarity were used with the two sets of attribute weight. Figure 7.5 shows the similarity matrices, similarity functions, similarity ranges, and taxonomy tree used to define degrees of similarity among the values of the attributes presented earlier in Table 7.4. For more explanations about these similarity measures, refer to section 5.4.1. Table 7.13 shows the adjusted weight and role of each of these attributes.

Highway Class	Express	National	Regional	Collector
National	0.2			
Regional	0	0.8		
Collector	0	0.6	0.8	
Local	0	0.4	0.6	0.8

Region	Western	Central	Eastern
Central	0		
Eastern	0	0.8	
Northern	0.2	0.6	0.4

Structural System	Simple
Continuous	0.4

Wearing Surface	Cement Mix
Bitominous Mix	0.2



Attribute	Symm	Relative	Interval	Rate
ADT	Yes	Yes	0.25	1
Truck %	Yes	Yes	0.25	1
Skew Angle	Yes	Yes	0.25	1
Beam Spacing	Yes	Yes	0.25	1
Age	Yes	Yes	0.25	1
MCR	Yes	No	1	1

Span Range	Relative	Similarity
R1	0 - 0.05	1
R2	0.05 - 0.10	0.8
R3	0.10 - 0.25	0.4
R4	> 0.25	0

Figure 7.5: Degrees of similarity used in the system validation

No.	Domain Entity Type or Time Dependent Set Type	Attribute	Role	Adjusted Weight
1	Bridge	Region	Discriminating	0.8
2		Highway Class	Discriminating	0.6
3		ADT	Discriminating	0.8
4		Truck%	Discriminating	0.2
5	Bridge Inspection	Inspection Date	Non-Discriminating	N.A.
6	Deck	Span Length	Discriminating	0.4
7		Skew Angle	Discriminating	0.0
8		Material	Indexing	0.0
9		Total Width	Non-Discriminating	N.A.
10		Beams Spacing	Discriminating	0.4
11		Structural System	Discriminating	0.6
12		Wearing Surface	Discriminating	0.4
13		Construction Date	Non-Discriminating	N.A.
14		No. of Beams	Non-Discriminating	N.A.
15	Girder	Material	Discriminating	0.6
16	Element Condition	MCR	Exact Matching	1.3
17		Age	Discriminating	0.2

Table 7.13: Adjusted attribute weights used in the system validation

To determine whether the predicted deck condition provided by the system is a good estimate of the true condition, three methods that have been reported by Ardit and Tokdemir (1999) are used. Each of these methods makes use of the overall case similarity of each retrieved case. These methods are as follows:

i. Method 1: The true condition is compared to the condition of the retrieved case that has the highest overall case similarity (no retrieval threshold is used).

ii. Method 2: The true condition is compared to the most frequent condition in the top ten retrieved cases, or fewer if ten are not available, that have an overall case similarity greater than or equal to 0.75 (retrieval threshold). Because deck conditions are represented as continuous float numbers, the condition range (from 1 to 6) is divided into groups (group width equal to 0.25) and the group that contains the highest number of retrieved cases is considered.

iii. Method 3: The true condition is compared to the average condition of the top five retrieved cases, or fewer if five are not available, that have an overall case similarity greater than or equal to 0.75 (retrieval threshold). The average of the predicted conditions is weighted using the overall case similarities to magnify the importance of the retrieved cases that have higher similarities.

Table 7.14 shows the results of testing the developed system with the two groups of cases (Group 1 and Group 2) and using both equal and adjusted attribute weights. These results are obtained for a prediction period equal to 2.4 years (the average inspection period in the cases of the two groups). This prediction period was used for validation purpose only, while the system can work with more practical periods such as 5 or 10 years. A tolerance value equal to 0.25, which represents 5% on the rating scale used that varies from 1 to 6, is used because deck conditions represent the aggregation of MCR of different deck portions and cannot be presented as integers. This tolerance is defined as the maximum acceptable absolute difference between the true condition and the predicted condition, which is determined depending on the user needs and the rating system used.

	Equal Weights				Adjusted Weights						
Testing Group	**Group 1**		**Group 2**		**Group 1**		**Group 2**		**Average**		
Calculation Method	% Retrieved	% Correct	% Retrieved	% Correct	% Retrieved	% Correct	% Retrieved	% Correct	% Retrieved	% Correct	% Retrieved Correct
1	100.0%	70.0%	100.0%	83.3%	100.0%	76.7%	100.0%	90.0%	100.0%	80.0%	80.0%
2	60.0%	61.1%	70.0%	85.7%	76.7%	78.3%	83.3%	80.0%	72.5%	76.3%	55.3%
3	60.0%	83.3%	70.0%	85.7%	76.7%	78.3%	83.3%	84.0%	72.5%	82.8%	60.0%
Average	**73.3%**	**71.5%**	**80.0%**	**84.9%**	**84.4%**	**77.7%**	**88.9%**	**84.7%**	**81.7%**	**79.7%**	**65.1%**
% Retrieved Correct	**52.2%**		**67.8%**		**65.6%**		**75.6%**				

Table 7.14: Testing results for equal and adjusted attribute weights

In Table 7.14, the percentage of query cases that obtained retrieved cases, whether with correct or incorrect solutions are listed in the "% Retrieved" column, while the percentage of retrieved cases that obtained correct solutions are listed in the "% Correct" column. These percentages are listed for both Group 1 and Group 2 using each of the three calculation methods presented earlier and for both equal and adjusted weight sets. For example: in Group 2 with Equal weights, the calculation method 3 obtained retrieved cases for 25 query cases out of the 30 cases used (83.3%) and correct solutions for 21 query cases out of these 25 cases (84.0%). The average of the "% Retrieved" (e.g. 88.9% for Group 2 with adjusted weights), the average of the "% Correct" (e.g. 84.7% for Group 2 with adjusted weights), and the average of the "% Retrieved" multiplied by the "% Correct", which is called "% Retrieved Correct", (e.g. 75.6% for Group 2 with adjusted weights), are calculated for the three calculation methods and presented for each testing group with both equal and adjusted attribute weights. The "% Retrieved Correct" represents the percentage of the query cases that obtained retrieved cases with correct solutions.

The effect of adjusting attribute weights is manifested in the increase of the average percentage of retrieved correct solutions from 52.2% to 65.6% for the Group 1 (used for training), and confirmed by the increase of the average of correct solution from 67.8% to 75.6% in the Group 2 (used for testing). This increase represents the effect of the retrieval knowledge on the results of the CBR system. Although acquiring retrieval knowledge from the domain experts may improve the obtained results, the process of knowledge acquisition has not been carried out because it is outside the scope of this study.

The average of the percentages of retrieved and correct solutions varies according to the calculation method used. This variation is shown in the "Average" column in Table 7.14. Although Method 3 fails sometimes to give any solution (% Retrieved 72.5%), it provides the highest percentage of correct solutions out of the retrieved cases (82.5%). Method 1 has the highest percentage of retrieved cases (100%) because it retrieves the solution of the best-matched cases regardless of its degree of similarity. Method 2 always has the same percentage of retrieved cases as Method 3, because of using the same retrieval threshold, but it has a lower percentage of correct solutions (76.3%).

In order to provide an interval estimate for the percentage of retrieved correct solutions, the Group 2 is used as a random sample with a size "n" equal to 30 and a proportion "p" equal to 75.6%. The population proportion "π" is estimated as follows:

$$\pi = p \pm Z_{\frac{\alpha}{2}} \sqrt{\frac{p(1-p)}{n}}$$

Where,

α	Level of significance
Z	Standard normal distribution parameter (valid for $n \geq 30$)

Applying this equation with α = 0.05 (i.e. confidence interval 95%) results in an interval estimate for the population proportion ranging from 60% to 90%. This estimate represents the accuracy of the CBR model in predicting the condition of bridge decks. It should be noted that such an estimate of the model accuracy has not been carried out for any of the deterioration models presented earlier in Chapter 2.

To evaluate this accuracy, a stepwise regression procedure was employed to build a multiple non-linear regression model using the same set of independent variables presented earlier in section 7.3 as the entry variables. Additional variables were considered by transforming the existing quantitative variables into non-linear expressions using different mathematical functions, such as square, cube, logarithm, square root, etc. Variables were screened by choosing a minimum value of 3.0 for the F-statistic of their test of hypothesis and by selecting the variables that produce the highest coefficient of determination and F-statistic (iterative process). Using the subset of data utilized in building the CBR case library and testing the CBR model, the regression model shown in Table 7.15 was built.

Predictor	Coefficient	Standard Deviation	t-ratio	p-value
Constant	0.074870	0.025270	2.96	0.00
Span Length^3	-0.000002	0.000001	-3.43	0.00
Beam Spacing^2	0.005328	0.002245	2.37	0.02
Deck MCR^4	0.000134	0.000026	5.17	0.00
Express Highway	-0.080780	0.021570	-3.75	0.00
Simple Structure System	-0.048540	0.015560	-3.12	0.00
Northern Region	-0.085330	0.018240	-4.68	0.00
Western Region	-0.078580	0.019020	-4.13	0.00
Cement Wearing Surface	-0.066620	0.025400	-2.62	0.01
Precast Material	0.060110	0.022470	2.67	0.01
Prestressed Material	-0.220730	0.059870	-3.69	0.00

S= 0.1325 R-Sq = 12.9% R-Sq(adj) = 11.1%

Analysis of Variance

Source	DF	SS	MS	F	P
Regression	10	1.23957	0.12396	7.06	0
Residual Error	477	8.3734	0.01755		
Total	487	9.61297			

Table 7.15: Results of stepwise regression model

Allowing the same tolerance value (i.e. 0.25), which represents the maximum amount of error that the user accepts in the predicted conditions, the regression model provided a point estimate for the population proportion equals 40% compared to 75.6% for the CBR model. Figure 7.6 shows a comparison between the results obtained from the CBR and regression models for different tolerances. This figure indicates that the CBR model has a better accuracy than the regression model especially when low tolerance values are used.

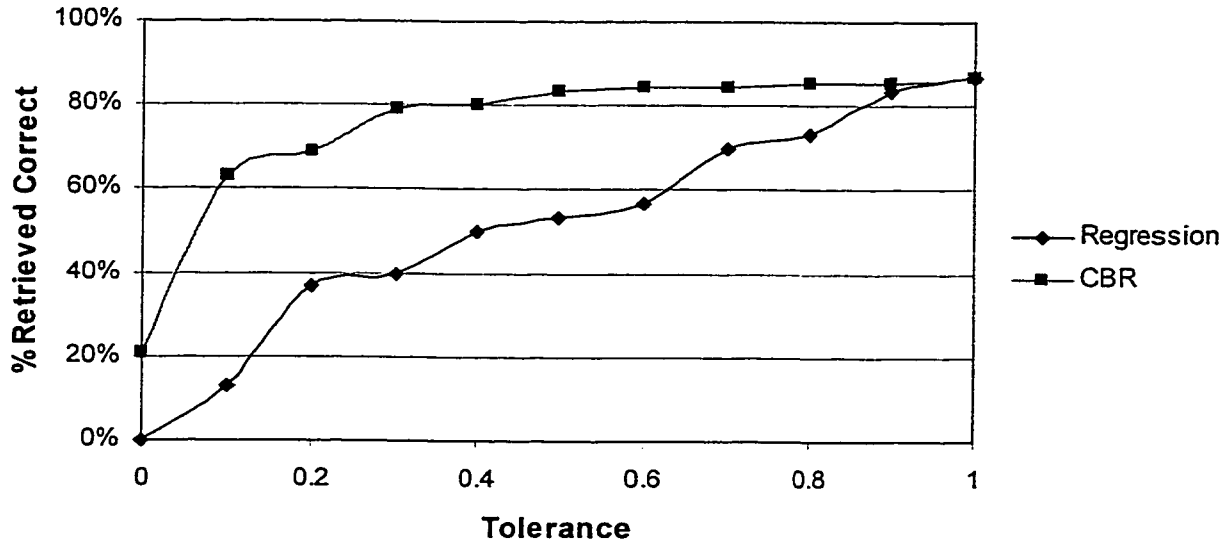


Figure 7.6: Comparing the accuracy of CBR and regression models

In order to study the sensitivity of the testing results with respect to the method of calculation and adjusting the attribute weights, percentages of retrieved correct solutions were computed using every calculation method at different tolerance values with both equal and adjusted attributes in Group 2 as shown in Figures 7.7 and 7.8. This sensitivity analysis indicated that calculating the "% Retrieved Correct" using Methods 2 and 3 is greatly affected by adjusting the attribute weights, while Method 1 is slightly affected. This is because Methods 2 and 3 have a restriction on the retrieved case similarities,

which may often result in no retrieved cases when the attribute weights are not adjusted. Also, it can be observed that the "% Retrieved Correct" increases by increasing the tolerance up to a certain value, then, it remains almost constant. This value is around 0.5 for equal weights and 0.4 for adjusted weights.

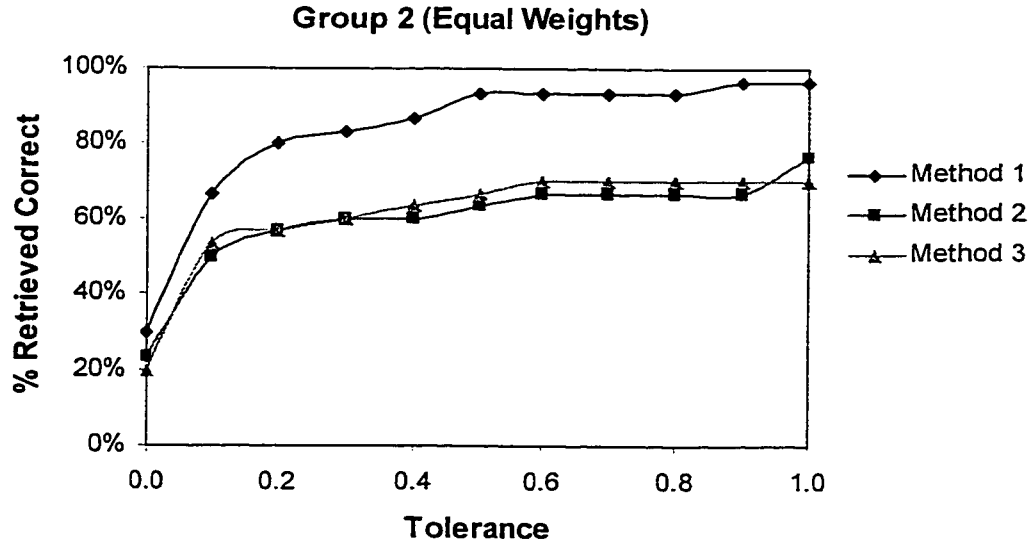


Figure 7.7: Percentage of retrieved correct solutions (for group 2 with equal weights) at different values of tolerance and using different calculation methods

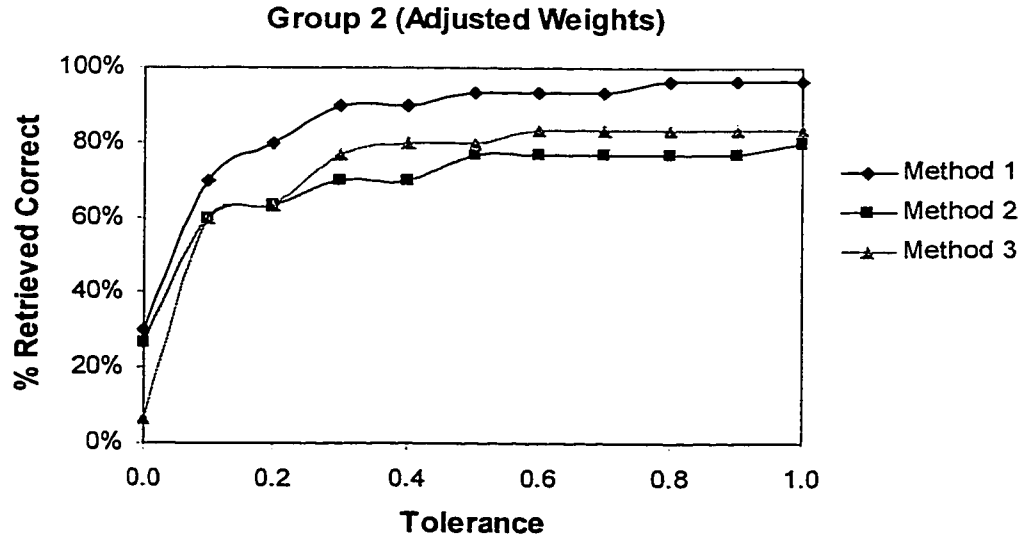


Figure 7.8: Percentage of retrieved correct solutions (for group 2 with adjusted weights) at different values of tolerance and using different calculation methods

Attribute weights significantly influences the accuracy of the retrieval process, thus, the sensitivity of this process to the weight of each attribute has to be measured (Kriegsman and Barletta 1993). In the developed system, only the sensitivity of the percentage of correct solutions to the weight of the "Deck MCR" attribute was measured. This is because this attribute is the most important one (i.e. has the highest weight) and it represents the importance of matching the condition history to matching other case features. This is sufficient since the number of attributes involved in the retrieval process is large and the same procedures can be followed in measuring the sensitivity to the weight of any of them. Figure 7.9 shows the results of this sensitivity analysis for Group 1 using different calculation methods. It can be concluded that increasing the deck MCR weight increases the percentage of correct cases up to a certain value depending on the method of calculation. In average, the "% Retrieved Correct" continued to increase by increasing the deck MCR weight up to a value of 1.3, then it began to decrease. This justifies the selection of a time-dependent set weight equal to 1.3 when adjusting attribute weights.

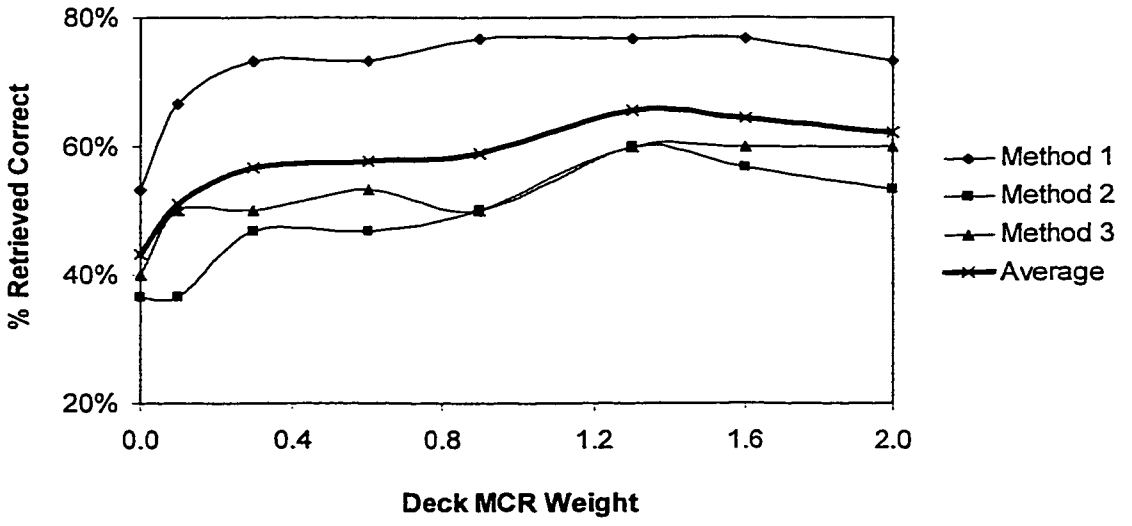


Figure 7.9: The sensitivity of testing results to deck MCR weight

Although evaluating system efficiency is an important part of validating any CBR system, it is difficult to carry out this evaluation quantitatively (Simoudis 1992). This is due to the absence of empirical data on running time and storage capacity of other systems that were used in solving the same problem using the same cases. Nevertheless, these data were recorded for the developed system that was running on a Pentium II 266 MHz processor. It was found that the time required for retrieving cases for 30 query cases is about 30 sec., and the space required for storing the data of 259 cases is about 3.0 MB.

7.5 SYSTEM EVALUATION

The developed system has to be evaluated based on the research objectives and system requirements presented earlier in Chapters 2 and 3 respectively. With respect to the practicality objectives, the system was able to work with the bridge data that are currently available in the MTQ database in spite of its inadequacy and incompleteness. Although the system was also able to import data from the MTQ database to build its case library using the ODBC capability, the full compatibility with the other BMS modules (e.g. optimization and cost modules) was not tested. This compatibility is expected to be achieved easily since the developed system can be used as an embedded application in an integrated system such as the BMS. The ability of the system to be updated easily was tested through the normal addition of new cases and new data in its case library. Because of the unavailability of maintenance data, the capability of performing "what-if" analysis for different maintenance decisions was also not tested. However, the system is able to store the maintenance history of bridge cases and to retrieve the bridge cases that had specific maintenance decisions.

With respect to the accuracy objectives, the CBR model was able to incorporate all the available factors that affect bridge deck deterioration; These factors include the effect of past conditions on future conditions and recognize the interaction between a bridge deck and its supporting girders. The CBR model provided a higher accuracy than the regression model developed using the same data. This performance has been evidenced through the percentage of correct cases resulted from each model: 75.6% for the CBR model and 40% for the regression model (with tolerance equals 0.25). These results have been obtained in the absence of bridge deck improvement actions and with limited bridge deck condition history. The availability of such data would improve the performance of the CBR model rather than the regression model because historical data can not be incorporated into the regression model. It is also important to report that the performance of the CBR model has not been compared against the performance of Markovian models because of two reasons: first, Markovian models are restricted to discrete integer condition ratings and can not be used with aggregated condition ratings; second, the large number of explanatory variables used in the validation will lead to segmenting the data sets into several segments, each with a few number of data points and, consequently, unreliable transition probabilities.

With respect to the versatility objectives, the CBR model was able to handle bridge decks with varying inspection period. Although the system was only tested for modeling the deterioration of concrete decks in beam bridges and using the condition rating system of Quebec (from 1 to 6), the system has the ability to model the deterioration of any bridge component regardless of the construction material, bridge type, and rating system.

The satisfaction of the following system requirements was presented above while discussing the achievement of the research objectives: representation of bridge component interactions (requirement # 3 in section 3.3.1), accumulation of bridge data (requirement # 8), system compatibility (requirement # 11), and system versatility (requirement # 12). Although the system is able to represent cases with complex hierarchical decomposition (requirement # 1), it was tested only using cases with a simple hierarchical decomposition that consists of three domain entity types (i.e Bridge, Deck, and Girder). The representation of time-dependent data requirement (requirement # 2) was fulfilled through the use of time-dependent sets, such as bridge inspection and element condition, that act as containers for time-dependent attributes. This type of containers can also be utilized in representing bridge maintenance actions. Reusing case attributes, sharing attribute values, and deriving data requirements (requirements # 4 and 9) were all satisfied and tested. Examples are: reusing the "Material" attribute in both "Deck" and "Girder" domain entity types, sharing the values of the "Region" attribute among the three domain entity types, and deriving the values of the "Beam Spacing" attribute from the values of the "Total Width" and "No. of Beams" attributes. The versatile similarity measures (requirement # 5) supported by the system were all employed in the developed application and provided efficient ways for calculating attribute similarities. Although the system has the ability to represent attribute weights and similarities as fuzzy numbers (requirement # 6), and to utilize the inference engine of CLIPS in adapting the retrieved cases (requirement # 7), these capabilities were not tested because the required knowledge is unavailable. The system extensibility requirement (requirement # 13) was satisfied because the system supports the addition of new domain

entity types to the domain model (i.e. extending case structure) and new attributes to the existing domain entity types (i.e. extending case contents). The system modularity requirement (requirement # 10) was fulfilled in implementing the CBR shell CBRMID as presented earlier in Chapter 6.

Testing the developed CBR system has brought up some drawbacks of using the CBR approach in solving the problem of modeling bridge deterioration. These drawbacks can be summarized as follows:

1. Bridge components with new construction materials, special structural systems, or unique inspection history may not find similar cases in the CBR case library. This can be a serious limitation for transportation agencies that have small databases and few inspection and maintenance records.
2. CBR systems require domain-specific knowledge (i.e. retrieval knowledge and adaptation knowledge) that has to be acquired from domain experts or literature. The process of knowledge acquisition is cumbersome and time-consuming. Moreover, there is no formal method that was developed for acquiring this knowledge, which may result in subjective and biased opinions.
3. The subjectivity of bridge condition assessment, the heterogeneity of bridge data, and the existence of unobserved/unrecorded deterioration factors may lead to an inconsistent case library that significantly affects the retrieved solutions.

CHAPTER 8

SUMMARY, CONTRIBUTIONS, AND RECOMMENDATIONS

8.1 SUMMARY

Bridge deterioration models have become an integral part of any advanced BMS. The role of these models is to predict the future condition of different bridge components and to estimate their remaining service life. This prediction is necessary for determining optimal maintenance actions and future funding needs. Therefore, this research focused on developing a bridge deterioration model that provides the state-of-the-art BMS's with accurate predictions of the future condition of different bridge components.

The literature review of bridge deterioration models has revealed four types of models: mechanistic, deterministic, stochastic, and artificial intelligence. Mechanistic models are efficient for modeling a specific deterioration mechanism at the project level and cannot be used for modeling bridge deterioration at the network level. Deterministic models are efficient for modeling bridge deterioration at the network level using a mathematical or statistical formulation such as Straight-line Extrapolation, Stepwise Regression, Linear and Non-linear Regression, B-spline Approximation, or Constrained Least Squares. Stochastic models use different methodologies to account for the uncertainty of the deterioration process such as Probability Distribution, Markovian models, and Simulation techniques. Artificial intelligence (AI) models use Artificial Neural Networks (ANN) to predict future bridge conditions. Deterministic, stochastic, and ANN models neglect the

effect of past bridge conditions and maintenance actions on the predicted condition, disregard the interaction among different bridge components, and segment the bridge population into groups of explanatory variables. Moreover, these models are difficult to update when new data are obtained or different rating systems are used.

Therefore, the case-based reasoning (CBR) approach, which is categorized as an AI model, is introduced to provide BMS's with a realistic, accurate, and generic deterioration model that eliminates the shortcomings of the state-of-the-art models. The basic idea of using CBR in modeling bridge deterioration assumes the similarity in the performance of different bridges that have similar physical features (e.g. material, structural systems, cross section, span, etc.), environmental and operating conditions (e.g. traffic, region, highway class, etc.), and inspection and maintenance history. This assumption was justified through the development of a "proof of concept" CBR system that supports: the accumulation of bridge data, the representation of time-dependent data, the hierarchical decomposition of bridges, the interaction among different bridge components, the variation of similarity measures, case adaptation, data derivation, data reuse, system versatility, system compatibility, and system extensibility.

Investigating the current commercial CBR shells revealed that none of these shells could be employed to generate a CBR system with the above-mentioned capabilities. Therefore, a decision was made to develop a CBR shell from scratch using the object-oriented programming language C++ along with the support of the object-oriented database management system (OODBMS) ObjectStore 6.0. This shell was designed to

allow the user to construct a domain model that describes concepts, data, and relationships of any infrastructure domain to model the deterioration of its facilities. For modeling the deterioration of bridges, a domain model was built by decomposing bridges hierarchically into seven levels of granularity (massing, systems, sub-systems, assemblies, sub-assemblies, and elements), identifying bridge data (static and dynamic), and establishing relationships among bridge concepts.

The developed CBR shell consists of three main modules: Case Based Reasoner, for case retrieval and adaptation; System Maintainer, for case and knowledge accumulation and modification; and Database for persistent data storage. The design model of the developed shell was presented in four stages, each of which deals with one aspect of the CBR system. In case representation, the hierarchical decomposition of domain entity types and their time-dependent set types were used to depict the domain model, while the hierarchical decomposition of domain entities and their time-dependent sets were used to represent case structures. Attributes and their values were represented as separate objects to allow the system to be extended and modified. In case retrieval, retrieval knowledge such as attribute weights and similarity measures were represented inside attribute objects. The nearest neighbor approach was employed to build the retrieval algorithm. In case adaptation, knowledge was represented as IF-THEN rules that can be manipulated by the inference engine of the expert system programming language CLIPS. In case of knowledge accumulation and modification, maintenance and repair operations are provided by the system maintainer to keep the system updated and consistent.

To implement the developed design model, the Rapid Data Development (RDD) tools of ObjectStore were used to generate the coding of the entity classes in C++. Control and boundary classes were implemented in seven separate projects by using the Microsoft Visual C++ 6.0 and Microsoft Foundation Classes (MFC). Compiling and linking each of these projects resulted in a software component that has its own functionality and corresponds to one module or sub-module in the CBR shell. Three other ready-made components were used to access the database for both read and write operations, allow data transfer from any ODBC data source, and run the inference engine of CLIPS. An overview of the developed prototype was presented with an illustrative example.

In order to test the developed system, inventory and inspection data of 240 Quebec bridges were used. These data contain the condition rating of bridge decks along with the factors that affect bridge deck deterioration. A statistical analysis was employed to check the consistency and completeness of the data and to obtain knowledge from it. In this analysis, the associations between the deterioration rate and several explanatory variables were measured by using coefficients of correlation for numeric variables and analysis of variance tests for symbolic variables. The multiple linear regression model developed using this data indicated that the available data are inadequate and incomplete to represent the true associations. Two data groups, each containing about 30 bridge decks, were used for training and testing the developed system. Testing results showed that the developed system could provide 75.6% correct solutions in average, while the regression model developed using the same data could provide 40% correct solutions.

8.2 CONTRIBUTIONS

The contributions of this research are grouped into three areas: modeling bridge deterioration, representing bridge data, and developing a CBR shell. Below is a highlight on the contributions made in each of these areas.

With respect to modeling bridge deterioration, a new approach that employs the use of CBR to predict the future condition and estimate remaining service life of a network of bridges was introduced. CBR is an AI approach that works with the specific knowledge encapsulated in previous cases and can incorporate domain knowledge with the reasoning process to refine the retrieved solution. This approach has eliminated most of the shortcomings of deterministic, stochastic, and ANN models presented in Chapter 2. CBR has contributed to solving the problems of bridge deterioration models as follows:

1. CBR retrieves not only bridge components that have the same physical features as the query case, but also those that have similar inspection and maintenance history. This means that the effect of the current condition on the predicted condition (neglected in the deterministic models) is considered and the state dependency (neglected in the Markovian models) is no longer neglected. Moreover, matching the condition history minimizes the effect of the uncertainty and randomness of the deterioration process, which reduce the need for stochastic approaches.
2. CBR is able to retrieve similar cases regardless of the amount of data used to describe these cases. This means, all the available explanatory variables that affect the deterioration process are considered as case attributes and involved in the reasoning process with no need for the segmentation technique. This technique is

used by most of the deterministic and stochastic models to account for the effect of some explanatory variables. This technique has a negative effect on the outcome of these models (refer to section 2.3.2).

3. CBR is not restricted to a constant inspection period or discrete condition states as in Markovian models. The use of interpolation in the retrieval algorithm allows the comparison of cases with variable inspection periods and aggregated conditions. In addition, CBR can retrieve cases regardless of the rating system used in representing their conditions as long as the rating system is consistent or there is a transformation mechanism among the different rating systems.
4. CBR does not apply only to modeling the deterioration of complete bridge structures but it applies also to modeling the deterioration of different bridge components at any level of complexity (massings, systems, sub-systems, assemblies, sub-assemblies, elements, etc.). This contribution diminishes the need for many deterioration models, each concerned with only one type of bridge components, and allows modeling the deterioration of complicated components. Furthermore, the CBR approach can be utilized in modeling the deterioration of infrastructure facilities other than bridges. It should be noted that this approach was tested for modeling the deterioration for bridge decks (uncomplicated cases) only.
5. The CBR model is updated as a natural by-product to the accumulation of new cases, inspection records, and maintenance actions. This update is not sophisticated as in deterministic and stochastic models. Furthermore, the accuracy of the CBR model continues to be improved with the update, whereby the coverage of the case library increases and the possibility of finding similar cases becomes higher.

6. The CBR model considers the interaction among the deterioration mechanisms of different bridge components while modeling their deterioration with no need for additional models to represent these interactions.
7. The CBR model provides a higher level of accuracy than the corresponding regression model. This performance was proved by testing the two models using actual cases; the CBR model provided correct solutions in 75.7% of cases, while the regression model provided correct solutions in 40% of cases.

With respect to representing bridge data, a new approach for representing bridges in BMS's was introduced. This approach has the following contributions:

1. A bridge is represented as a collection of interrelated domain entities, each domain entity is a container that stores the information about one bridge component. These domain entities are arranged in a hierarchical decomposition that breaks down bridge components based on different criteria, such as location and function. This hierarchical decomposition represents bridge components at different levels of granularity, such as bridge massing, bridge system, bridge sub-system, bridge assembly, bridge sub-assembly, and bridge element, that serve as an efficient abstraction mechanism.
2. Bridge data, either static or dynamic, can be stored at any level of a bridge hierarchical decomposition providing various levels of details in data storage.
3. Time-dependent bridge data are grouped and represented in separate containers called time-dependent sets. These containers can be shared by many domain entities and can be decomposed into more detailed sets.

4. Bridge attributes are represented in separate containers that are defined for each category of domain entities (i.e. domain entity type) and each category of time-dependent sets (i.e. time-dependent set type). Attribute values are also represented in separate containers that are connected with their relevant attributes. This representation allows:
    i. storing information about the attributes themselves,
    ii. reusing attributes by different domain concepts,
    iii. adding, removing, and updating attributes at run-time without losing the values stored for these attributes, and
    iv. sharing attribute values among different domain entities or time-dependent sets, which saves storage space.
5. Bridge axes and spans are represented with containers called location identifiers that are linked with the domain entities of the bridge hierarchical decomposition. These identifiers determine the location of domain entities and facilitate the transformation to other bridge representations.
6. Association relationships among domain entities are introduced to represent the interaction of bridge components.

With respect to developing a CBR shell, a prototype of a shell called CBRMID was developed to satisfy the requirements of modeling bridge deterioration and bridge data representation. CBRMID provides decision-makers in the respective agencies with a generic tool that can be used in predicting the future condition of any infrastructure

facility, and consequently, in optimizing their MR&R decisions. This shell has contributed to the state-of-the-art CBR shells as follows:

1. CBRMID facilitates robust case representation with all the above mentioned features of bridge data representation.
2. Attribute weights and degrees of similarity can be represented as fuzzy numbers that reflect the way the domain experts recognize the relative importance of attributes and the relative similarity among their values.
3. Versatile similarity measures, such as similarity function, similarity ranges, similarity matrix, and taxonomy tree, are provided to calculate the similarity for different attribute types (i.e. continuous, segmented, enumerated, and hierarchical).
4. The conventional taxonomy tree of hierarchical attributes has been modified to allow repeating attribute values under different parent nodes, decomposing nodes based on a list of identified criteria, and calculating similarity among attribute values according to their matched criteria. These modifications have changed the taxonomy tree to be more realistic.
5. CBRMID supports matching cases with complicated and incompatible hierarchical decompositions. It also facilitates matching the historical record of the attribute used for prediction with recognizing the relative importance of the newer records over the older ones.
6. Case adaptation capabilities (manual and automatic) are supported in CBRMID through the use of the inference engine of the expert system programming language CLIPS 6.10 and the representation of IF-THEN rules in the CBRMID adaptation

knowledge base. The inference engine is also able to manipulate any additional rules implemented in the CLIPS syntax.

7. CBRMID facilitates an efficient case accumulation through the use of case templates and data transfer from any ODBC data sources. These capabilities speed up the creation of new cases and avoid having erroneous case structures.

8.3 RECOMMENDATIONS FOR FUTURE RESEARCH

In order to enhance the capabilities of the CBR approach in modeling bridge deterioration, the following points need to be explored in the future research:

1. Testing the CBR model with cases that have complex hierarchical decomposition, long inspection history, and past maintenance actions.
2. Validating the CBR model by operating it in a real-world environment where users can evaluate the system performance in various and real situations.
3. Evaluating the CBR model against some of the state-of-the-art deterioration models, such as Markovian models and ANN models. These models should be developed using the same data in order to provide unbiased evaluations.
4. Integrating the CBR model with the other modules of the host BMS, such as the cost and optimization modules. This integration is necessary for BMS's in order to perform their functionality.
5. Acquiring domain-specific knowledge through extensive interviews with bridge experts to obtain sufficient knowledge for case adaptation and to verify the retrieval knowledge used.

In order to upgrade the developed CBR shell (CBRMID) to industrial strength for daily use in real applications, some features demand more investigation, others require extension, and new features need to be added. Below are the features of the developed shell that are recommended for future research:

1. The domain model should be able to support reusing child domain entity types, such as Wall, among different parent domain entity types, such as Pier and Abutment. This feature eliminates the redundancy in defining the domain concepts and results in a more efficient domain model.
2. The adaptation knowledge base of CBRMID has to be extended to make the best use of the powerful features of CLIPS in representing and processing its rules.
3. Data derivation capabilities need to be expanded to support more complicated derivation equations.
4. The equation used in calculating the change of weight of the attribute used for prediction need to be modified to support different user needs.
5. The efficiency of CBRMID in terms of running time and storage capacity has to be evaluated with large infrastructure networks and huge amount of facility data.
6. The system GUI needs to be refined, so it becomes friendlier and more intuitive, and provides the user with help tips and warnings, and speed up the process of data entry with less possibility of errors.
7. New features that automate the generation of attribute weights from the cases stored in the case library are required to reduce the subjectivity of assigning these weights manually.

REFERENCES

1. Abed-Al-Rahim, I. J. and Johnston, D. W. (1995) "Bridge Element Deterioration Rates", Transportation Research Record, TRB, Vol. 1490, pp. 9-18.

2. Aktan, A. E., Farhey, D. N., Brown, D. L., Dalal, V., Helmicki, A. J., Hunt, V. J., and Shelley, S. J. (1996) "Condition Assessment For Bridge Management", ASCE, Journal of Infrastructure Systems, Vol. 4, No. 3, pp. 108-117.

3. Althoff, K., Auriol, E., Barletta, R., and Manago, M. (1995) "A Review of Industrial Case-Based Reasoning Tools", AI Perspective Report, AI Intelligence.

4. American Association of State Highway and Transportation Officials (AASHTO) (1993) "Guidelines For Bridge Management Systems", Washington, D.C.

5. American Association of State Highway and Transportation Officials (AASHTO) (1999) "Pontis: The Complete Bridge Management System", [Web Page], accessed on October 1999, http://aashtoware.camsys.com/html/PontisReport.html.

6. Amirbekian, V., Chendynski, T., Rastabiga, Z., and Ukowski, O. (1997) "Comparison of Eight OODBMS's", [Web Page], accessed on May 1999, http://galaxy.uci.agh.edu.pl/~vahe/products.htm.

7. Arditi, D. and Tokdemir, O. (1999) "Comparison of Case-Based Reasoning and Artificial Neural Networks", Journal of Computing in Civil Engineering, ASCE, Vol. 13, No. 3, pp. 162-169.

8. Ben-Akiva, M. and Gopinath, D. (1995) "Modeling Infrastructure Performance and User Costs", Journal of Infrastructure Systems, ASCE, Vol. 1, No. 1, pp. 33-43.

9. Ben-Akiva, M., Humplick, F., Madanat, S., and Ramaswamy, R. (1993) "Infrastructure Management under Uncertainty: Latent Performance Approach", Journal of Transportation Engineering, ASCE, Vol. 119, No. 1, pp. 43-58.

10. Bergmann, R. (1998) "On the Use of Taxonomies for Representing Case Features and Local Similarity Measures", Proceeding of the 6th German Workshop on Case-Based Reasoning (GWCBR'98), Berlin, Germany, March, pp. 23-32.

11. Bergmann, R. and Stahl, A. (1998) "Similarity Measures in Object-Oriented Case Representation", Proceedings of the European Workshop on Case-Based Reasoning (EWCBR'98), Dublin, Ireland, September, pp. 25-36.

12. Bergmann, R., Wilke, W., Vollrath, I., and Wess, S. (1996) "Integrating General Knowledge with Object-Oriented Case Representation and Reasoning", 4th German Workshop: Case-Based Reasoning – System Development and Evaluation, Berlin, Germany, March, pp. 120-127.

13. Bertino, E. and Martino, M. (1993) "Object-Oriented Database Systems: Concepts and Architectures", Addison-Wesley Publishers.

14. Booch, G., Jacobson, I., and Rumbaugh, J. (1997) "The UML Specification Documents", Rational Software Corp.

15. Botten, N. Kusiak, A., and Raz, T. (1989) "Knowledge Bases: Integration, Verification, and Partitioning", European Journal of Operational Research, Amsterdam, Netherlands, Vol. 42, pp. 111-128.

16. Boussabina, A. H. (1996) "The Use of Artificial Neural Networks in Construction Management: A Review", Construction Management and Economics, Vol. 14, pp. 427-436.

17. Bulusu, S. and Sinha, K. C. (1997) "Comparison of Methodologies to Predict Bridge Deterioration", Transportation Research Record, TRB, Vol. 1597, pp. 34-42.

18. Busa, G. D. (1985) "Modeling Concrete Bridge Deck Deterioration", Master of Science in Transportation, Massachusetts Institute of Technology, June.

19. Butt, A. A., Shahin, M. Y., Feighan, K. J., and Carpenter, S. H. (1987) "Pavement Performance Prediction Model Using the Markov Process", Transportation Research Record, TRB, Vol. 1123, pp. 12-19.

20. Cady, P. D. and Weyers R. E. (1984) "Deterioration Rates of Concrete Bridge Decks", Journal of Transportation Engineering, ASCE, Vol. 110, No. 1, pp. 34-44.

21. Carrier, E. and Cady, P. D. (1973) "Deterioration of 249 Bridge Decks", Transportation Research Record, TRB, Vol. 423, pp. 46-57.

22. Cesare, M. A., Santamarina, C., Turkstra, C., and Vanmarcke E. H. (1992) "Modeling Bridge Deterioration With Markov Chains", Journal of Transportation Engineering, ASCE, Vol. 118, No. 6, pp. 820-833.

23. CLIPS (1998a) "Clips Reference Manual, Volume I, Basic Programming Guide, Version 6.10", [Web Page], http://www.ghgcorp.com/clips/

24. CLIPS (1998b) "Clips Reference Manual, Volume II, Advanced Programming Guide, Version 6.10", [Web Page], http://www.ghgcorp.com/clips/

25. Collins, L. (1972) "An Introduction To Markov Chain Analysis", CATMOG, Geo Abstracts Ltd., University of East Anglia, Norwich.

26. Czepiel, E. (1995) "Bridge Management Systems; Literature Review and Search", [Web Page], accessed on May 1998, http://iti.acns.nwu.edu/pubs/tr11.html

27. DeStefano, P. D., and Grivas, D. A. (1998) "Method for Estimating Transition Probability in Bridge Deterioration Models", Journal of Infrastructure Systems, ASCE, Vol. 4, No. 2, pp. 56-62.

28. Dzeng, R. and Tommelein, I. (1995) "Case-Based Scheduling Using Product Models", Proceedings of the 2nd Congress on Computing in Civil Engineering, ASCE, Atlanta, Georgia, June, pp. 163-170.

29. Edlund, B., Lofqvist, P., and Johansson, P. (1994) "Case-Based Reasoning in Bridge Design", EG-SEA-AI 1st Workshop, Application of Artificial Intelligence in Structural Engineering, Lausanne, March, pp. 60-61.

30. Elmasri, R. and Navathe, S. (1994) "Fundamentals of Database Systems", The Benjamin/Cummings Publishing Company.

31. Federal Highway Administration (FHWA) (1995) "Recording and Coding Guide for the Structural Inventory and Appraisal Guide of the Nation's Bridges", Washington D. C., December.

32. Federal Highway Administration (FHWA) (1997) "Highway Statistics 1997", [Web Page], http://www.fhwa.dot.gov/ohim/hs97/hs97page.htm.

33. Flemming, U. (1994) "Case-Based Design in the SEED System", Knowledge-Based Computer-Aided Architectural Design, Elsevier, pp. 69-91.

34. Fowler, M. and Scott, K. (1997) "UML distilled: applying the standard object modeling language", Addison-Wesley Publishers.

35. Freyermuth, C. L., Klieger, P., Stark, D. C., and Wenke, H. N. (1970) "Durability of Concrete Bridge Decks-A Review of Cooperative Studies", Transportation Research Record, TRB, Vol. 328, pp. 50-60.

36. Goodman, M. (1989) "CBR in Battle Planning", Proceedings of a Workshop in CBR (DARPA'89), Pensacola Beach, Florida, May-June, pp. 312-316.

37. Hannus, M., Karstila, K., and Seren, K. (1995) "Generic Product Data Model for Product Data Exchange – Requirements, model and implementation" Proceeding of the 6th International Conference on Computing in Civil and Building Engineering, Berlin, Germany, July, pp. 283-290.

38. Haque, M. (1997) "Uniform Bridge Element Identification System For Database Management of Roadway Bridges", Journal of Bridge Engineering, ASCE, Vol. 2, No. 4, pp. 183-188.

39. Heister, F. and Wilke, W. (1998) "An Architecture for Maintaining Case-Based Reasoning Systems", Proceedings of the European Workshop on Case-Based Reasoning (EWCBR'98), Dublin, Ireland, September, pp. 221-232.

40. Honshu-Shikoku (1994) "Maintenance Management of Long-Span Bridges", Maintenance Department, Honshu-Shikoku Bridge Authority, Japan.

41. Howard, H. C., Abdalla, J. A., and Phan, D. H. D. (1992) "Primitive-Composite Approach for Structural Data Modeling ", Journal of Computing in Civil Engineering, ASCE, Vol. 6, No. 1, pp. 19-40.

42. Hua, G. B. (1996) "Residential Construction Demand Forecasting Using Economics Indicators: A Comparative Study of Artificial Neural Networks and Multiple Regression", Construction Management and Economics, Vol. 14, pp. 25-34.

43. Hudson, S. W., Carmichral III, R. F., Moser, L. O., and Hudson, W. R. (1987) "Bridge Management Systems", TRB – NCHRP, Report 300.

44. Jaczynski, M. and Trousse, B. (1998) "An Object-Oriented Framework for the Design and the Implementation of Case-Based Reasoners", 6th German Workshop on Case-Based reasoning, Berlin, Germany, March.

45. Jiang, Y. and Sinha, K. C. (1989) "Bridge Service Life Prediction Model Using the Markov Chain", Transportation Research Record, TRB, Vol. 1223, pp. 24-30.

46. Jiang, Y., Saito, M., and Sinha, K. C. (1988) "Bridge Performance Prediction Model Using the Markov Chain", Transportation Research Record, TRB, Vol. 1180, pp. 25-32.

47. Johnston, D. W., Chen, C., and Abed-Al-Rahim, I. (1994) "Developing User Costs for Bridge Management Systems", Transportation Research Circular, TRB, Vol. 423, pp. 139-149.

48. Kayser, J. R. and Nowak, A. S. (1989) "Capacity Loss due to Corrosion in Steel-Girder Bridges", Journal of Structural Engineering, ASCE, Vol. 115, No. 6, pp. 1525-1537.

49. Kolodner, J. (1993) "Case-Based Reasoning", Morgan Kaufmann Publishers, Inc.

50. Kriegsman, M. and Barletta, R. (1993) "Building a Case-Based Help Desk Application", IEEE Expert, Vol. 8, No. 6, pp. 18-26.

51. Kriviak, G. (1999) "Bridge Management System Development for Municipal Sized Inventories in Western Canada", Proceedings of 8th International Conference on Bridge Management, TRB - FHWA, Denver, Colorado, April.

52. Kueng, P. (1994) "Comparison of Ten OODBMS's", Swiss Computer Magazine, Vol. 6, pp. 60-63.

53. Kumar, H. and Krishnamorthy, C. (1995) "A Framework for case-based reasoning in engineering design", Artificial Intelligence for Engineering Design, Analysis, and Manufacturing, Cambridge University Press, Vol. 9, pp. 161-182.

54. Lafore, R. (1991) "Object-Oriented Programming in Turbo C++", Waite Group Press.

55. Larman, C. (1998) "Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design", Prentic Hall Inc.

56. Leake, D. B. (1996) "Case-Based Reasoning; Experiences, Lessons, and Future Directions", American Association for Artificial Intelligence.

57. Leinecker, R. and Archer, T. (1998) "Visual C++ 6.0 Bible", IDG Books Worldwide Inc.

58. Lipkus, S. E. (1994) "BRIDGIT Bridge Management Software", Transportation Research Circular, TRB, Vol. 324, pp. 43-54.

59. Lounis, Z. (1999) "A Stochastic and Multiobjective Decision Model for Bridge Maintenance Management", Infra 99 International Convention, Montreal, Quebec, November.

60. Madanat, S. and Ibrahim, W. H. W. (1995) "Poisson Regression Model for Computing Infrastructure Transition Probabilities", Journal of Transportation Engineering, ASCE, Vol. 121, No. 3, pp. 267-272.

61. Madanat, S., Karlaftis, M. G., and McCarthy, P. S. (1997) "Probabilistic Infrastructure Deterioration Models with Panel Data", Journal of Infrastructure Systems, ASCE, Vol. 3, No. 1, pp. 4-9.

62. Madanat, S., Mishalani, R., and Ibrahim, W. H. W. (1995) "Estimation of Infrastructure Transition Probabilities from Condition Rating Data", Journal of Infrastructure Systems, ASCE, Vol. 1, No. 2, pp. 120-125.

63. Maher, M., Balachandran, M., and Zhang, D. (1995) "Case-Based Reasoning in Design", Lawrence Erlbaum Associates, Inc.

64. Messner, J., Sanvido, V., Ikeda, M. (1994) "Developing an Object Based Planning System for Precast Concrete Building Structures", Computing in Civil Engineering, 1st Congress held in conjunction with A/E/C Systems, Washington, D.C., June, pp. 1426-1429

65. Ministry of Transportation of Ontario (MTO) (1991) "Ontario Structure Inspection Manual OSIM", Structural Office, St-Catherine, Ontario.

66. Ministry of Transportation of Quebec (MTQ) (1995) "Manual d'Inspection des Structures: Evaluation des Dommages", Bibliothèque Nationale du Québec, Gouvernement du Québec.

67. Ministry of Transportation of Quebec (MTQ) (1997) "Manual de l'Usage du Système de Gestion des Structures SGS-5016", Bibliothèque Nationale du Québec, Gouvernement du Québec.

68. Moore, D. S. and McCabe, G. P. (1993) "Introduction to the Practice of Statistics", W. H. Freeman and Company.

69. Mullins, C. S. (1994) "The Great Debate", Byte Magazine, Vol. 19, No. 4, April, pp. 85-96.

70. Ng, T. and Smith, N. J. (1998) "Verification and Validation of Case-Based Prequalification System", Journal of Computing in Civil Engineering, ASCE, Vol. 12, No. 4, pp. 215-226.

71. Object Design (1998a) "ObjectStore Rapid Database Development for PSE PRO for C++, Release 3.0 for Windows", Object Design Inc., November.

72. Object Design (1998b) "ObjectStore C++ API User Guide, Release 6.0, Beta 4", Object Design Inc., December.

73. Object Design (1999) "ObjectStore C++ Collections Guide and References, Release 6.0", Object Design Inc., March.

74. O'Brien III, D. E., Kohn, S. D., and Shahin, M. Y. (1983) "Prediction of Pavement Performance by Using Nondestructive Test Results", Transportation Research Record, TRB, Vol. 943, pp. 13-16.

75. Phan, D. and Howard, D. (1993) "The Primitive-Composite (P-C) Approach-A Methodology for Developing Sharable Object-Oriented Data Representations for Facility Engineering Integration", CIFE, Technical Report No. 85A, Stanford University, California, August.

76. POET (1998) "C++ SKD Programmer's Guide, Version 5.1", POET Software Corporation, San Mateo, California.

77. Quatrani, T. (1998) "Visual Modeling With Rational Rose and UML", Addison-Wesley Publishers Inc.

78. Ramarapu, N., Raisinghani, M. S., Frolick, M., and Prabhu, S., (1997) "Integration of Case-Based Reasoning with Object-Oriented Database Management Systems for Efficient Management of Large Case bases", [Web Page], accessed in June 1999, http://hsb.baylor.edu/ramsower/ais.ac.97/papers/ramarapu.html

79. Rankin, J., Froese, T., and Waugh, L. (1999) "Exploring the Application of Case-Based Reasoning to Computer-Assisted Construction Planning", Durability of Building Materials and Components 8, Institute of Research in Construction, Ottawa, Ontario, pp. 2526-2536.

80. Rivard, H. (1994) "Integration of the Building Design Process", Master of Applied Science, Concordia University, September.

81. Rivard, H. and Fenves, S. J. (2000) "A Representation for Conceptual Design of Buildings", Journal of Computing in Civil Engineering, ASCE, Vol. 14, No. 3, pp. 151-159.

82. Rivard, H., Fenves, S. J., and Gomez, N (1998) "Case-Based Reasoning for Conceptual Building Design", 1st International Conference on New Information Technologies for Decision Making in Civil Engineering, University of Quebec, Montreal, Quebec, October, pp. 355-366.

83. Roddis, W. M. K., and Bocox, J. (1997) "Case-Based Approach for Steel Bridge Fabrication Errors", Journal of Computing in Civil Engineering, ASCE, Vol. 11, No. 2, pp. 84-91.

84. Rumbaugh, J. (1997) "Models Through The Development Process", Journal of Object-Oriented Programming, NY: SIGS Publications, May.

85. Sanders, D. H. and Zhang, Y. J. (1994) "Bridge Deterioration Models for States with Small Bridge Inventories", Transportation Research Record, TRB, Vol. 1442, pp. 101-109.

86. Scherer, W. T. and Glagola, D. M. (1994) "Markovian Models for Bridge Maintenance Management", Journal of Transportation Engineering, ASCE, Vol. 120, No. 1, pp. 37-50.

87. Sen, R., Stroh, S., Olbinska, J., Hassiotis, S., and Mullins, G. (1999) "Development of a New Concept for Florida's Medium Span Bridges", [Web Page], accessed on November 1999, http://www.dot.state.fl.us/research-center/sfr845.html.

88. Shahin, M. Y. (1994) "Pavement Management for Airports, roads, and Parking Lots", Chapman & Hall.

89. Shahin, M. Y., Nunez, M. M., Broten, M. R., Carpenter, S. H., and Sameh, A. (1987) "New Techniques for Modeling Pavement Deterioration", Transportation Research Record, TRB, Vol. 1123, pp. 12-19.

90. Sianipar, P. R. M. and Adams, T. M. (1997) "Fault-Tree Model of Bridge Element Deterioration Due to Interaction", Journal of Infrastructure Systems, ASCE, Vol. 3, No. 3, pp. 103-110.

91. Simoudis, E. (1992) "Using Case-Based Retrieval for Customer Technical Support", IEEE Expert, Vol. 7, No. 5, pp. 7-12.

92. Sincich, T. (1994) "A Course in Modern Business Statistics", Macmillan College Publishing Company, Inc.

93. Slater, R. E., (1998) "FY2000 Budget in Brief", Report from the Secretary of Transportation, U.S. Department of Transportation.

94. Sobanjo, J. O. (1991) "A Decision Support Methodology for the Rehabilitation and Replacement of Highway Bridges", Ph.D. Dissertation, Texas A&M University, May.

95. Sobanjo, J. O. (1997) "A Neural Network Approach to Modeling Bridge Deterioration", Proceedings of the 4th Congress on Computing in Civil Engineering, ASCE, Philadelphia, PA, June, pp. 623-626.

96. Stanfill, C. (1988) "Learning to Read: A Memory-Based Model", Proceedings of a Workshop in CBR (DARPA'88), Morgan Kaufmann, California.

97. Tah, J. H. M., Carr, V., and Howes, R. (1999) "Information Modeling for Case-Based Construction Planning of Highway Bridge Projects", Advances of Engineering Software, Vol. 30, pp. 495-509.

98. Tanaka, S., Mikami, I., Yoneda, S., and Maeda, H. (1996) "An Expert System Using Case-Based Reasoning for Selecting Retrofitting Methods of Fatigue Damage on Steel Bridges", Proceedings of the 3rd conference on Bridge Management, E & FN Spon, London, Britain, pp. 17-25.

99. Tecinno (1999) "CBR-Works 4: Compendium", Techinno Gmbh, Kaiserslautern, Germany.

100. Tee, A., Bowman, M., and Sinha, K. (1988) "Application of Fuzzy Logic to Condition Assessment of Concrete Slab Bridges", Transportation Research Record, TRB, Vol. 1184, pp. 22-30.

101. Thompson, P. D. (1998) "A New Bridge Management System For Ontario", 1st International Conference on New Information Technologies for Decision Making in Civil Engineering, Ecole Technologie Supérieure, Université du Québec, Montreal, Quebec, October, pp. 1385-1396.

102. Thompson, P. D. and Shepard, R. W. (1994) "Pontis", Transportation Research Circular, TRB, Vol. 324, pp. 35-42.

103. Thompson, P. D., Small, E. P., Johnson, M., and Marshall, A. R. (1998) "The Pontis Bridge Management System", Structural Engineering International Journal, IABSE, Vol. 8, pp. 303-308.

104. Tokdemir, O. B., Ayvalik, C., and Mohammadi, J. (2000) "Prediction of Highway Bridge Performance by Artificial Neural Networks and Genetic Algorithms", Proceeding of the 17th International Symposium on Automation and Robotics in Construction (ISARC), September, Taipei, Taiwan, pp. 1091-1098.

105. Transport Canada (1998) "Transportation In Canada", Annual Report.

106. Turner, D. S. and Richardson, J. A. (1994) "Bridge Management System Data Needs and Data Collection", Transportation Research Circular, TRB, Vol. 423, pp. 5-15.

107. U.S. Department of Transportation (DOT) (1999) "Memorandum: Strategic Goals for Deficient Bridges", July.

108. U.S. Government (2000) "Budget of the United States Government Fiscal Year 2001", Office of Management and Budget, February.

109. Watson, I. (1997) "Applying Case-Based Reasoning: Techniques for Enterprise Systems", Morgan Kaufmann Publisher, Inc.

110. Watson, I., and Marir, F. (1994) "Case-Based Reasoning: A Review", [Web Page], accessed in May 1999, http://ai-cbr.org/classroom/cbr-review.html.

111. Wiegand, N. and Adams, T. (1995) "Establishing Relationships in an OODBMS", Proceedings of the 2nd Congress on Computing in Civil Engineering, ASCE, Atlanta, Georgia, June, pp. 1252-1258.

112. Yang, J. and Yau, N. (1996) "Applications of Case-Based Reasoning in Construction Engineering and Management", Proceedings of the 3rd Congress on Computing in Civil Engineering, ASCE, Anaheim, California, July, pp. 663-669.

APPENDIX A

Comparisons of CBR Tools and OODBMS's

List of Items:

Comparison of 4 CBR Tools

	Name	KATE Tools	ART*Enterprise	CBR-Works	The Easy Reasoner
Administrative Data	Contact	Michel Manago	Pat Mahaffey	Martin Braeuer	Doug Lauffer
	Company	AcknoSoft	Brightware Inc.	TecInno GmbH	The Haley Enterprise, Inc.
	Address	58 rue du Dessous-des-Berges	350 Ignacio Boulevard, Novato	Sauerwiesen 2	413 Orchard Street
		75013 Paris, France	CA94949, USA	67661 Kaiserslautern, Germany	Sewickley, PA 15143, USA
	Phone	(331) 44248800	800.532.2890	49 6301 606 36	(412) 741-6420
	Fax	(331) 44248866	415-884-4740	49 6301 606 67	(412) 741 -6457
	E-mail	manago@ibpc.fr	mahaffey@brightware.com	braeuer@tecinno.com	doug@haley.com
	Web Site	http://www.acknosoft.com	http://www.brightware.com	http://www.tecinno.com	http://www.haley.com
	Evaluation Copy	Not Available	Not Available	Download (3 months)	Not Available
	Price (USD)	2490 + Shipping	Free + 2500 Maintenance Fee	2500	2389 + Shipping
Technical Data	Suitability	Help desks, customer support, trouble shooting, complex diagnosis, data mining and other CBR applications.	Client-server MIS applications and other CBR applications	Trouble shooting, complex diagnosis, data mining, product selection and other CBR applications, particularly e-commerce solutions	Help desks, customer support, trouble shooting, complex diagnosis, data mining and other CBR applications.
	Platform	Windows 3.1, 95/98, NT.	Windows 95/98, NT, OS/2, Solaris	Windows NT, 95/98, OS/2, Apple Macintosh, Unix Systems of SUN, HP, SNI, SGI & Digital Alpha with Unix or Windows NT.	Windows 3.1, 95/98, NT and a range of UNIX platforms.
	Reasoning	Retrieval and adaptation	Retrieval and adaptation	Retrieval and adaptation	Retrieval and adaptation
	Case Representation	Object-Oriented model	Object-Oriented model	Object-Oriented model	Attribute-value pairs
	Case Retrieval	Nearest-Neighbor and Induction	Nearest-Neighbor	Nearest-Neighbor	Nearest-Neighbor
	Case Matching	Content-based matching	Content-based matching	Content-based and structure-based matching	Content-based matching
	Similarity Assessment	Customized similarity assessment	Fixed similarity assessment	Customized similarity assessment	Fixed similarity assessment
	Case Adaptation	No Adaptation	Functions and Rules	Functions and Rules	Functions and Rules
	Database Integration	Provide database connectivity	Provide database connectivity	Provide database connectivity	Provide database connectivity
	Embeddability	Embeddable as C code	Embeddable (API for C++)	Not embeddable	Embeddable within C++ applications
	Interface	Graphical User Interface	Graphical User Interface	Graphical User Interface	Graphical User Interface

Comparison of 8 OODBMS's (Amirbekian et al. 1997):

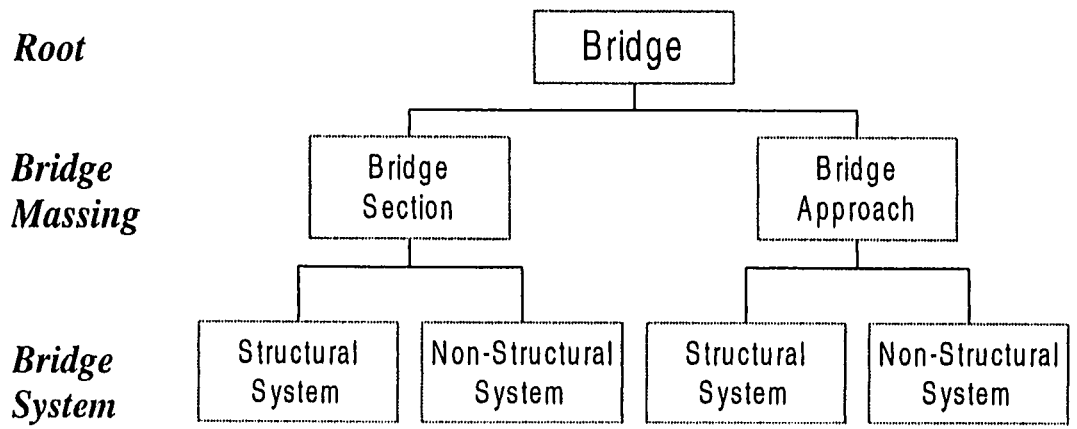
PRODUCT/ CRITERION	O2 (v.5.0)	OBJECTIVITY (v.5.0)	OBJECTSTORE (v.5.0)	VERSANT (v.5.0)	POET (v 5.0)	GEMSTONE	JASMINE	ITASCA v. 2.3.5
ADMINISTRATIVE DATA								
Company	O2 Technology	Objectivity, Inc.	Object Design, Inc.	Versant Object Technology	POET Software, Inc.	GemStome Systems, Inc.	Computer Associates International, Inc.	IBEX Object Systems, Inc.
Web Site	www.o2tech.com	www.objectivity.com	www.odi.com	www.versant.com	www.poet.com	www.gemstone.com	www.cai.com	www.ibex.ch
Price (US$)	3000$ per seat	3000$ per seat	3500$ per seat	NO DATA	$2500	NO DATA	NO DATA	NO DATA
BASIC TECHNICAL DATA								
DBMS supports user defined data types	YES	YES	YES	YES	YES	YES	YES	YES
DBMS supports IS_A relationships	YES	YES	YES	YES	YES	YES	YES	YES
DBMS supports PART_OF relationships	YES	YES	YES	YES	YES	YES	YES	YES
DBMS supports multiple inheritance	YES	YES	YES	YES	YES	Smalltalk-No Java-YES	YES	YES
DBMS checks the cardinality between objects	YES	YES	YES	YES	YES	NO	YES	YES
DBMS supports application programming in C++	YES	YES	YES	YES	YES	YES	YES	YES
STANDARDS								
DBMS supports ODMG C++ binding	YES, v1.2	YES, all basic capabilities, but not collections	YES, v1.2	YES, v1.2	YES	NO	YES, via TDI Inc.	NO DATA
DBMS supports the standard SQL in interactive mode	NO	YES	NO	YES	YES	Smalltalk - YES Java - NO	YES via ODBC	NO
DBMS supports the standard SQL in embedded mode	NO	YES	YES	YES, via ODBC	YES	Smalltalk - YES Java - NO	YES via ODBC	NO
DBMS supports a database language based on SQL	YES	YES	YES	YES	YES	Smalltalk-Yes Java-No	YES	YES

QUERIES								
DBMS supports ad-hoc queries with GUI	YES	YES,	YES	YES	YES	YES	YES	YES
DBMS supports ad-hoc queries with object-oriented language (e.g. C++)	YES	YES, as long as those tools support ODBC	YES	YES	YES	YES Java Smalltalk	YES	YES
SCHEMA UPDATE								
DBMS supports ad-hoc updates of the DB-schema with a GUI	YES	YES	YES	YES	YES	YES	YES	YES
DBMS supports ad-hoc updates of the DB-schema with an object-oriented language	YES	YES	YES	YES	YES	YES Java Smalltalk	YES	YES C++ Smalltalk
PLATFORM AND INTEGRATION								
DBMS is supported by an integrated object-oriented CASE-tool	YES	NO	YES RATIONAL ROSE	NO	YES RATIONAL ROSE, OEW	NO	YES	NO
An application running on the OODBMS can read data on other DBMS	YES	YES	YES	YES	YES DB2 MS-SQL	YES Sybase Informix	YES	NO
DBMS supports single-user multi-tasking environment	YES	YES	YES,	YES	YES	YES	YES	YES
DBMS supports MS-Windows	NT	NT and 95	NT and 95	NT and 95	NT and 95	NT	NT and 95	NT and 95

APPENDIX B
Hierarchical Decomposition of Bridges

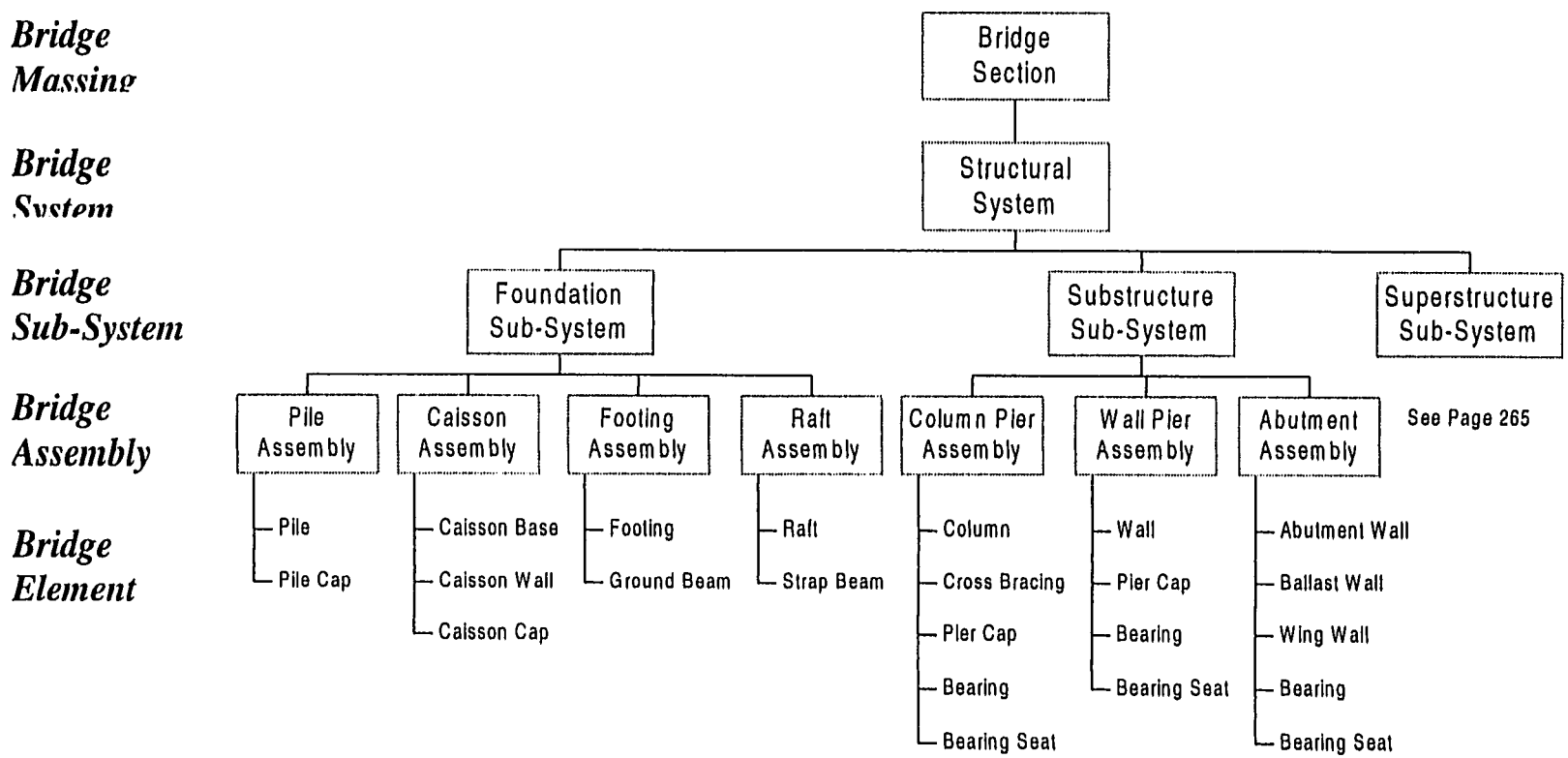
List of Items:

Decomposition of bridges to the level of bridge systems

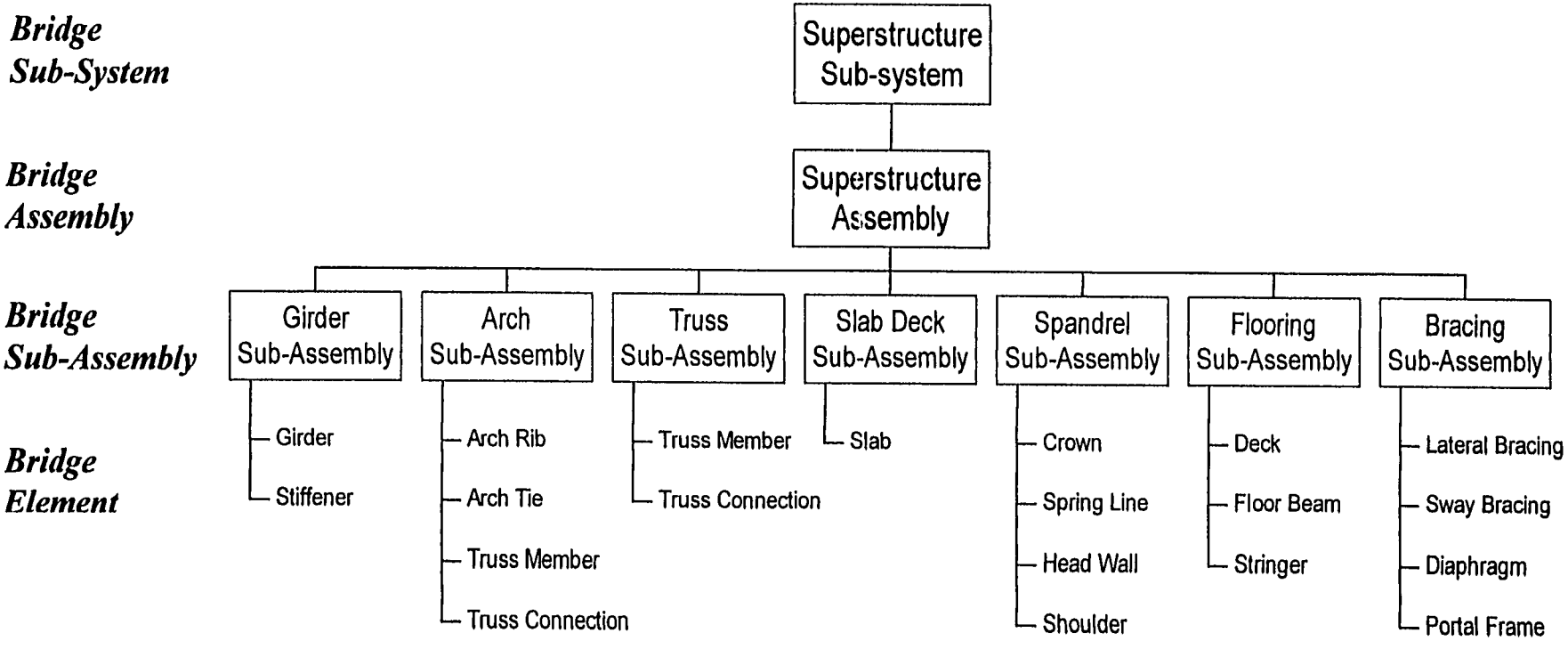


See Page 264 See Page 266 See Page 267 See Page 268

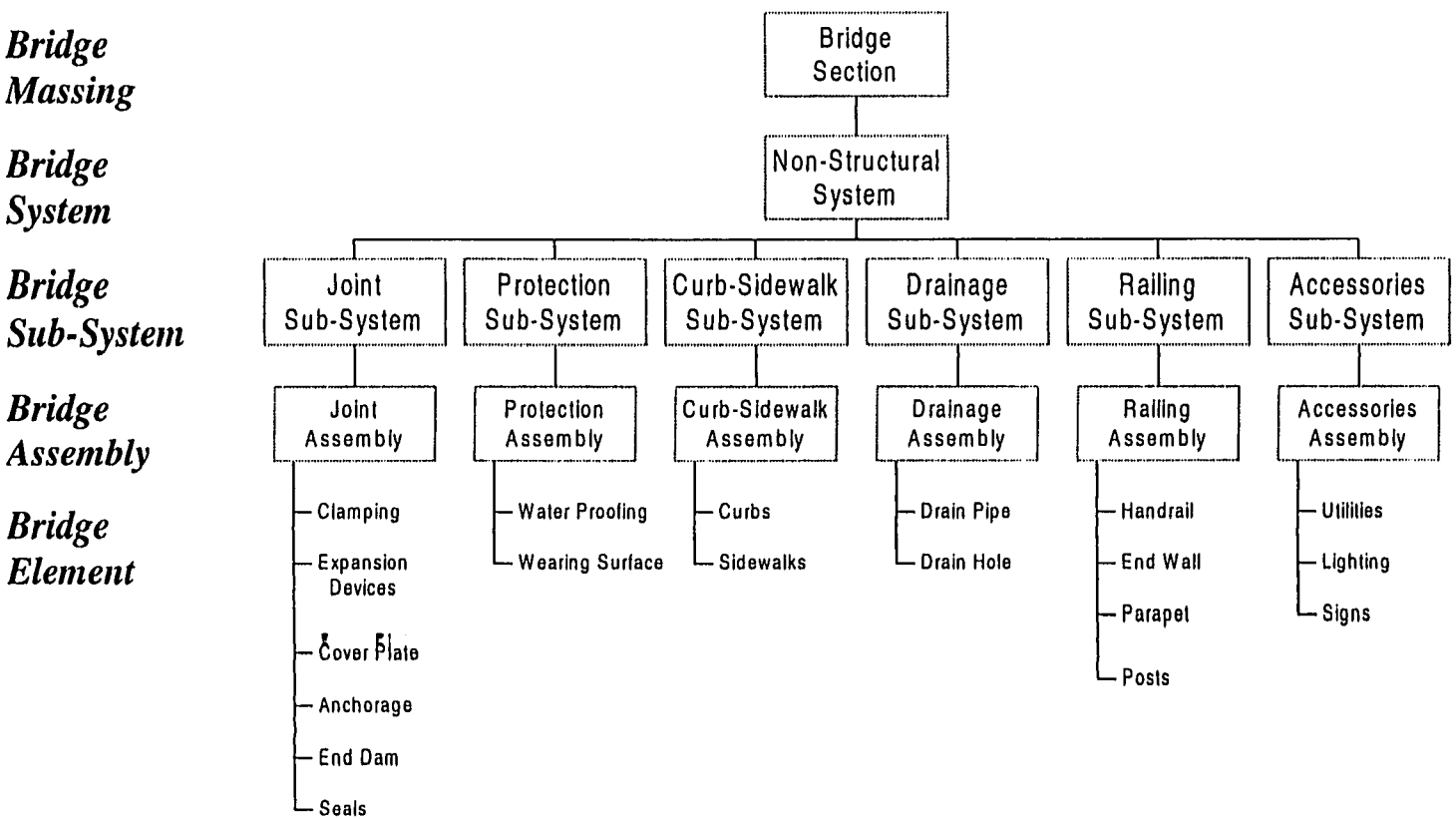
Decomposition of bridge section structural systems to the level of bridge elements



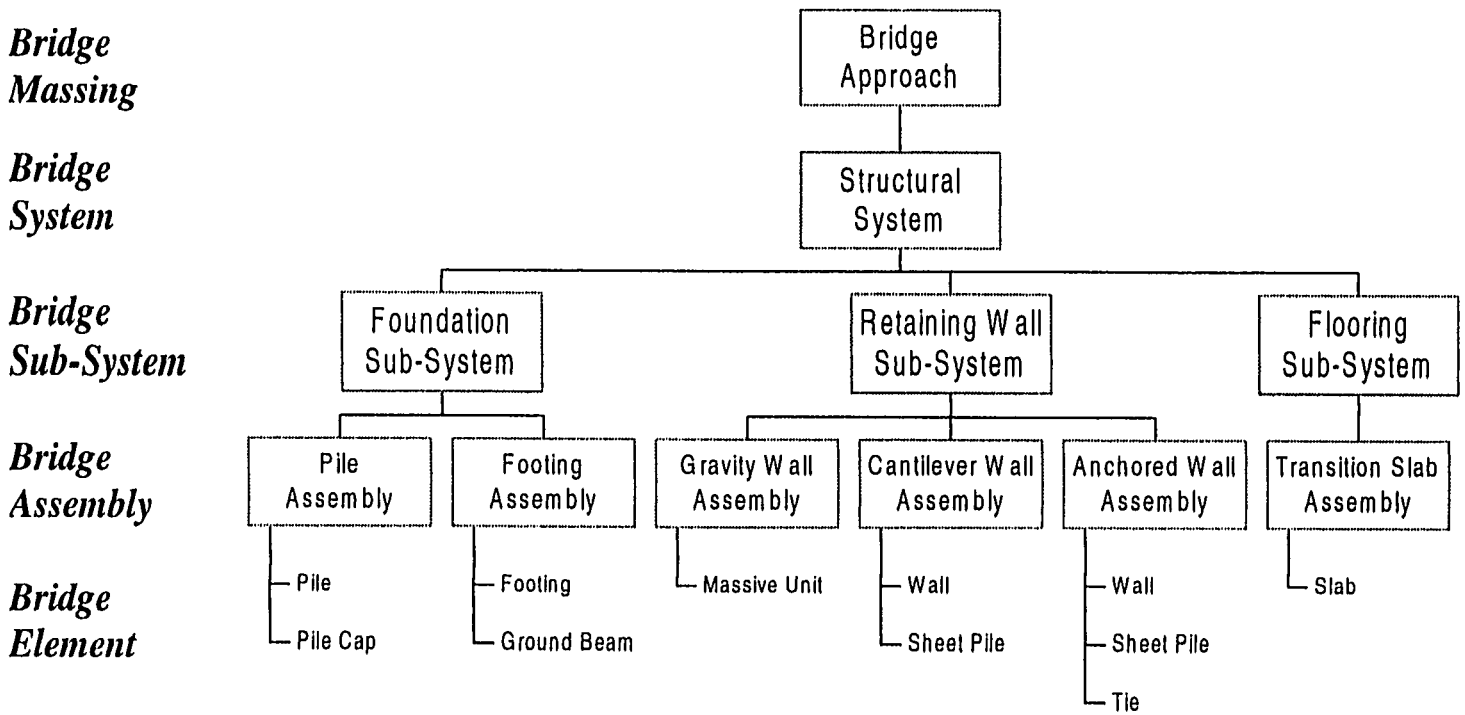
Decomposition of superstructure sub-systems to the level of bridge elements



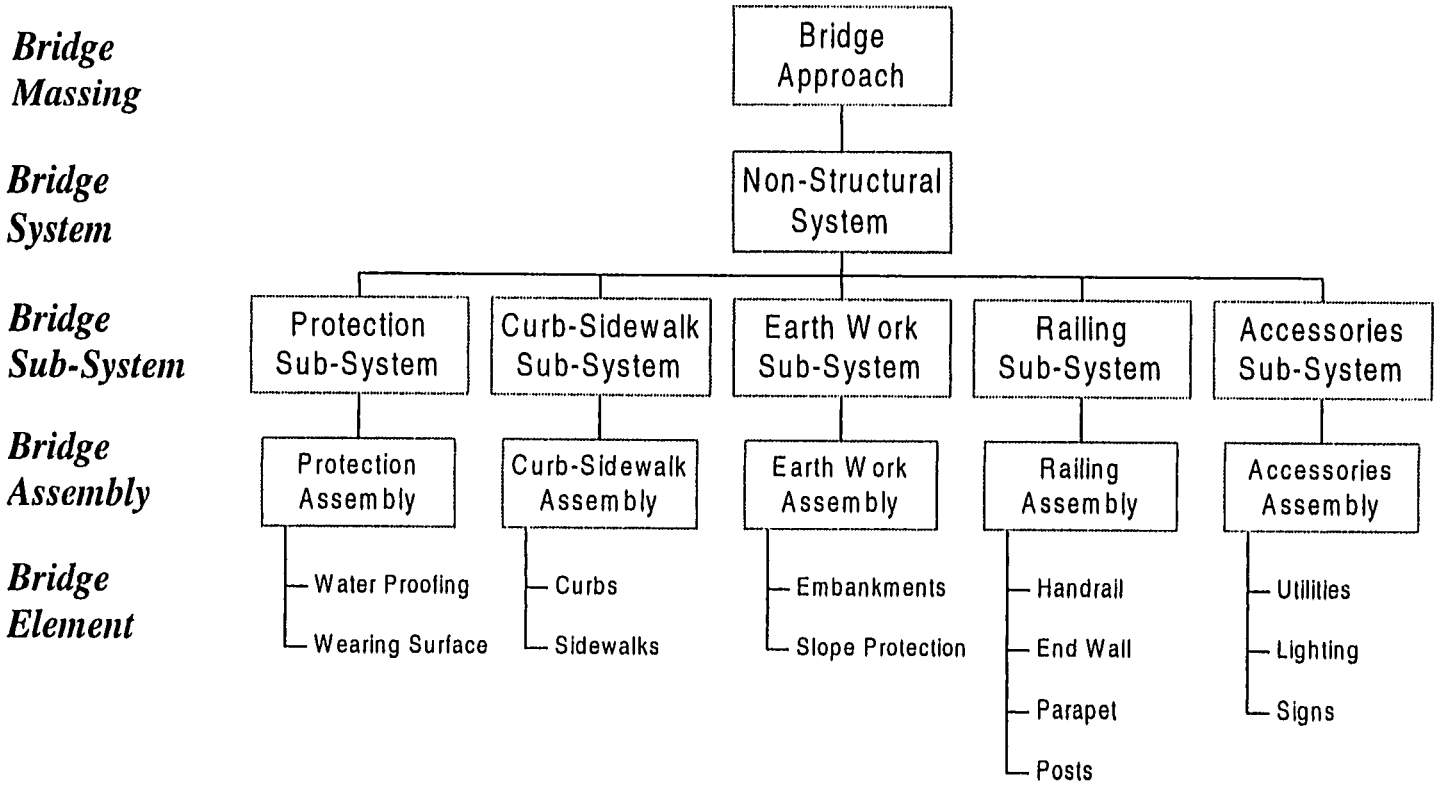
Decomposition of bridge section non-structural systems to the level of bridge elements



Decomposition of bridge approach structural systems to the level of bridge elements



Decomposition of bridge approach non-structural systems to the level of bridge elements



APPENDIX C
System Maintenance and Repair Operations

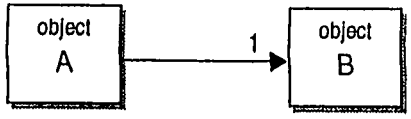
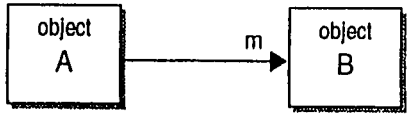
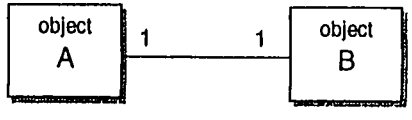
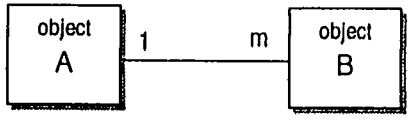
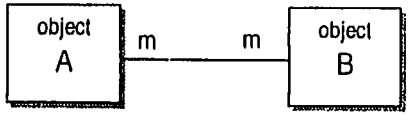
List of Items:

Maintenance Operations

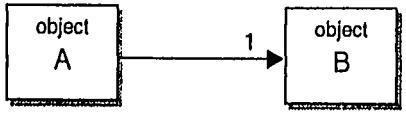
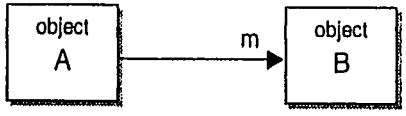
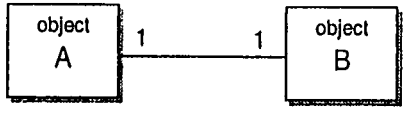
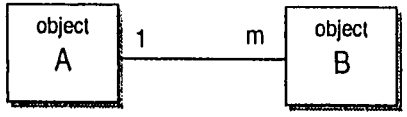
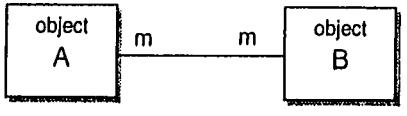
Database Sub-Module	Class	Maintenance Operation	Not Allowed When
Domain Model	Domain Entity Type	Add	Entity type name is null
			Entity type name is repeated
			Entity type is not a root and its parent is unspecified
		Remove	Entity type has any relations with other objects
		Update	Changing entity type name to null
			Changing entity type name to an existing one
			Changing the parent domain entity type
	Time-Dependent Set Type	Add	Set type name is null
			Set type name is repeated
			Parent domain entity type is unspecified
		Remove	Set type has any attributes, sets, or child set types
		Update	Changing set type name to null
			Changing set type name to an existing one
			Having the set type shared and reused at the same time
			Sharing a set type that does not belong to the parent entity type
Retrieval Knowledge Base	Attribute	Add	Attribute name is null
			Attribute name is repeated
			Attribute container is unspecified
			Attribute type is unspecified
			Having a time attribute or an attribute used for prediction in an entity type
			Having more than one time attribute in the same time-dependent set type
			Having more than one attribute used for prediction for the same entity type
		Remove	Re-used by other domain entity types or time-dependent set types
		Update	Changing attribute name
			Changing attribute type
			Changing attribute container
			Having the attribute shared and reused at the same time
			Sharing an attribute that does not belong to the parent container
			Having a time attribute or an attribute used for prediction in an entity type
			Having more than one time attribute in the same time-dependent set type
			Having more than one attribute used for prediction for the same entity type
	Fuzzy Number	Add	Fuzzy number name is null
			Fuzzy number name is repeated for the same variable
		Remove	Fuzzy number is used by other objects
		Update	Changing fuzzy number name
			Changing the variable of a fuzzy number
	Polynomial Term	Add	Term name is null
			Term name is repeated for the same attribute
			Parent attribute is not numeric and derived from the same or parent container
			Term attribute is unspecified
		Remove	Not applicable
		Update	Changing term name
			Changing the parent attribute
			Changing the term attribute
	Similarity Range	Add	Range name is null
			Range name is repeated for the same attribute
			Parent attribute is not a segmented attribute
		Remove	Not applicable
		Update	Changing range name
			Changing the parent attribute
	Matrix Row	Add	Row name is null
			Row name is repeated for the same attribute
			Parent attribute is not an enumerated attribute
		Remove	Not applicable
		Update	Changing row name
			Changing the parent attribute
	Tree Node	Add	Node name is null
			Node name is repeated for the same parent node
			Parent node is unspecified
		Remove	Not applicable
		Update	Changing node name to null
			Changing node name to an existing one
			Changing the parent node

Adaptation Knowledge Base	Adaptation Rule	Add	Rule id is null
			Rule id is repeated for the same domain entity type
			Parent domain entity type is unspecified
		Remove	Not applicable
		Update	Changing rule id
			Changing the parent domain entity type
	Rule Condition	Add	Condition id is null
			Condition id is repeated for the same rule
			Parent adaptation rule is unspecified
			Connective constraint is unspecified
			Left attribute and its container are unspecified
			Condition test is unspecified
		Remove	Not applicable
		Update	Changing condition id
			Changing the parent adaptation rule
			Changing left attribute or its container to unspecified
			Changing condition test is unspecified
			Changing connective constraint is unspecified
	Rule Action	Add	Action id is null
			Action id is repeated for the same rule
			Parent adaptation rule is unspecified
			Action type is unspecified
			Left attribute and its container are unspecified for assignment actions
		Remove	Not applicable
		Update	Changing action id
			Changing the parent adaptation rule
			Changing action type to unspecified
			Changing left attribute or its container to unspecified for assignment actions
Case Library	Domain Entity	Add	Entity id is null
			Entity id is repeated for the same parent domain entity
			Entity is not a root and its parent is unspecified
			Parent domain entity type is unspecified
		Remove	Not applicable
		Update	Changing entity id to an existing one under the same parent entity
			Changing the parent domain entity
			Changing the parent domain entity type
	Time-Dependent Set	Add	Set id is null
			Set id is repeated for the same parent entity and set type
			Parent time-dependent set type is unspecified
			Parent domain entity is unspecified
			Set type is not a root and the parent time-dependent set is unspecified
		Remove	Not applicable
		Update	Changing set id
			Changing the parent domain entity
			Changing the parent time-dependent set type
			Changing the parent time-dependent set
	Location Identifier	Add	Identifier name is null
			Identifier type is null
			Identifier name is repeated for the same parent entity type
			Parent domain entity is not a root
		Remove	Not applicable
		Update	Changing identifier name
			Changing identifier type to unspecified
			Changing the parent domain entity
Case Retrieval & Adaptation	Retrieval Operation	Add	Operation id is null
			Operation id is repeated for the same query case
			Query case is unspecified
		Remove	Not applicable
		Update	Changing operation id
			Changing the relevant query case

Repair Operations for Addition Transactions

Relationship	Transaction	Problem	Repair Operation
 The type: One-way relationship The multiplicity: (one or many) to one	A is added	A is not connected with B	Include B in the A constructor
	B is added	B is not connected with A	Assign B to A data member
 The type: One-way relationship The multiplicity: (one or many) to many	A is added	A is not connected with B	User insert member function of A
	B is added	B is not connected with A	User insert member function of A
 The type: Two-way relationship The multiplicity: one to one	A is added	A is not connected with B	Use set member function of A
	B is added	B is not connected with A	Use set member function of B
 The type: Two-way relationship The multiplicity: one to many	A is added	A is not connected with B	User insert member function of A
	B is added	B is not connected with A	Use set member function of B
 The type: Two-way relationship The multiplicity: many to many	A is added	A is not connected with B	User insert member function of A
	B is added	B is not connected with A	User insert member function of B

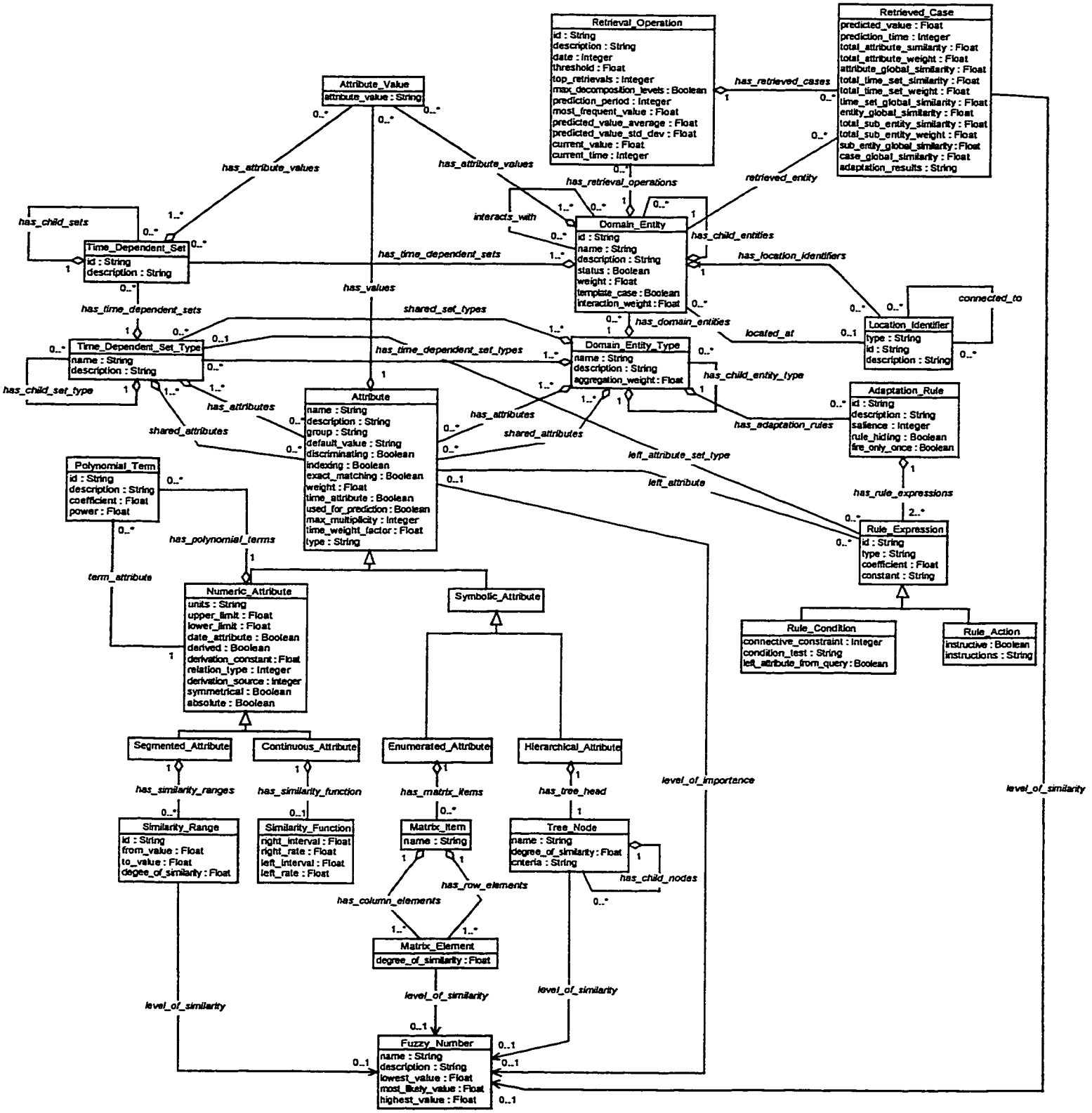
Repair Operations for Deletion Transactions

Relationship	Transaction	Problem	Repair Operation
 The type: One-way relationship The multiplicity: (one or many) to one	A is deleted	B will be deleted	Let A points to Null before deletion
	B is deleted	A will point to an non-existing object	Let A points to Null before deletion
 The type: One-way relationship The multiplicity: (one or many) to many	A is deleted	All B objects will be deleted	Empty A collection before deletion
	B is deleted	A will point to an non-existing object	Remove B from A collection before deletion
 The type: Two-way relationship The multiplicity: one to one	A is deleted	B object will remain with a Null relation	Delete B object if the relation is aggregation else no repair
	B is deleted	A object will remain with a Null relation	Delete B object if the relation is aggregation else no repair
 The type: Two-way relationship The multiplicity: one to many	A is deleted	B objects will remain with a Null relation	Delete B object if the relation is aggregation else no repair
	B is deleted	B will be removed from A collection automatically	No repair
 The type: Two-way relationship The multiplicity: many to many	A is deleted	A will be removed from B collection automatically	No repair
	B is deleted	B will be removed from A collection automatically	No repair

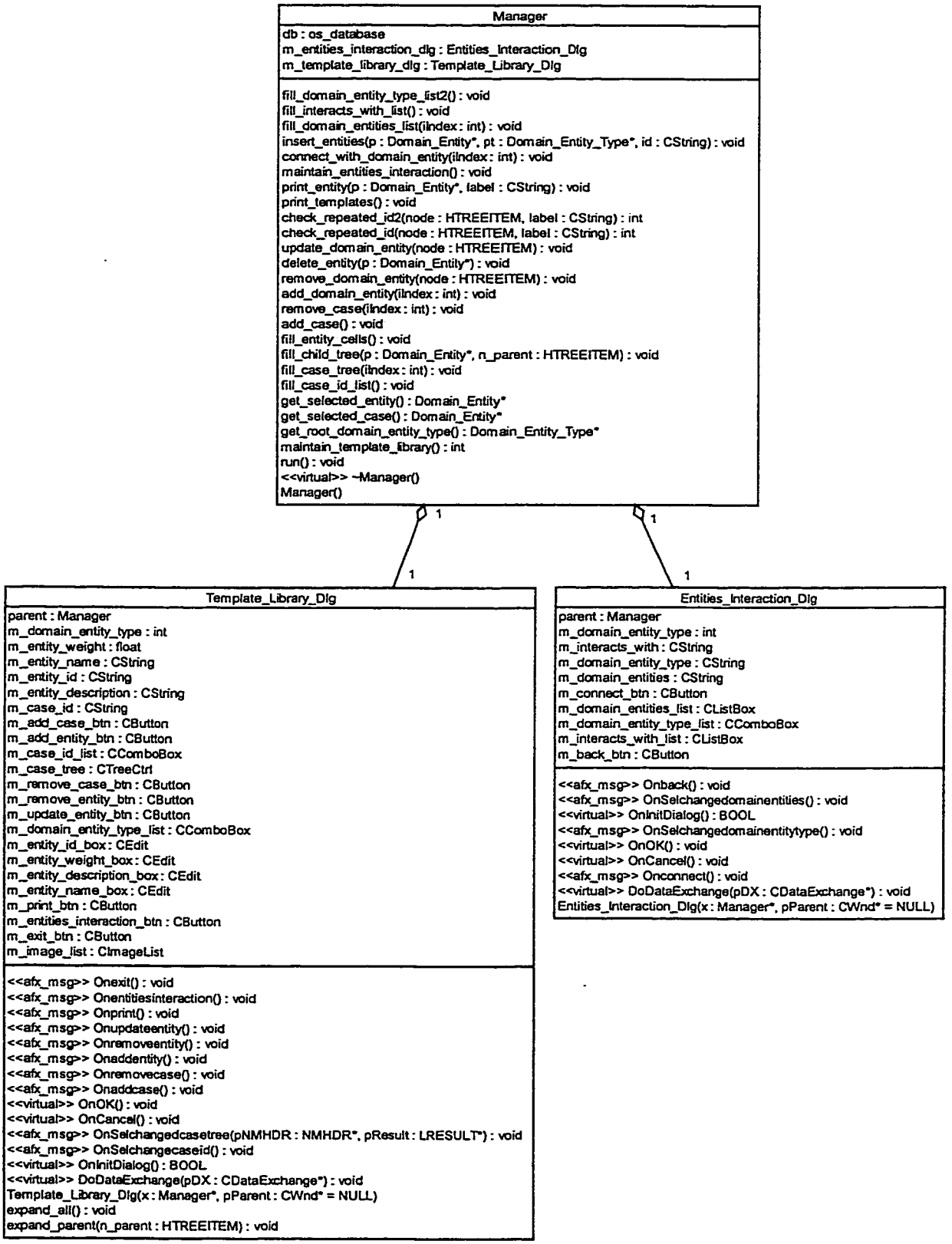
APPENDIX D
System Design Model

List of Items:

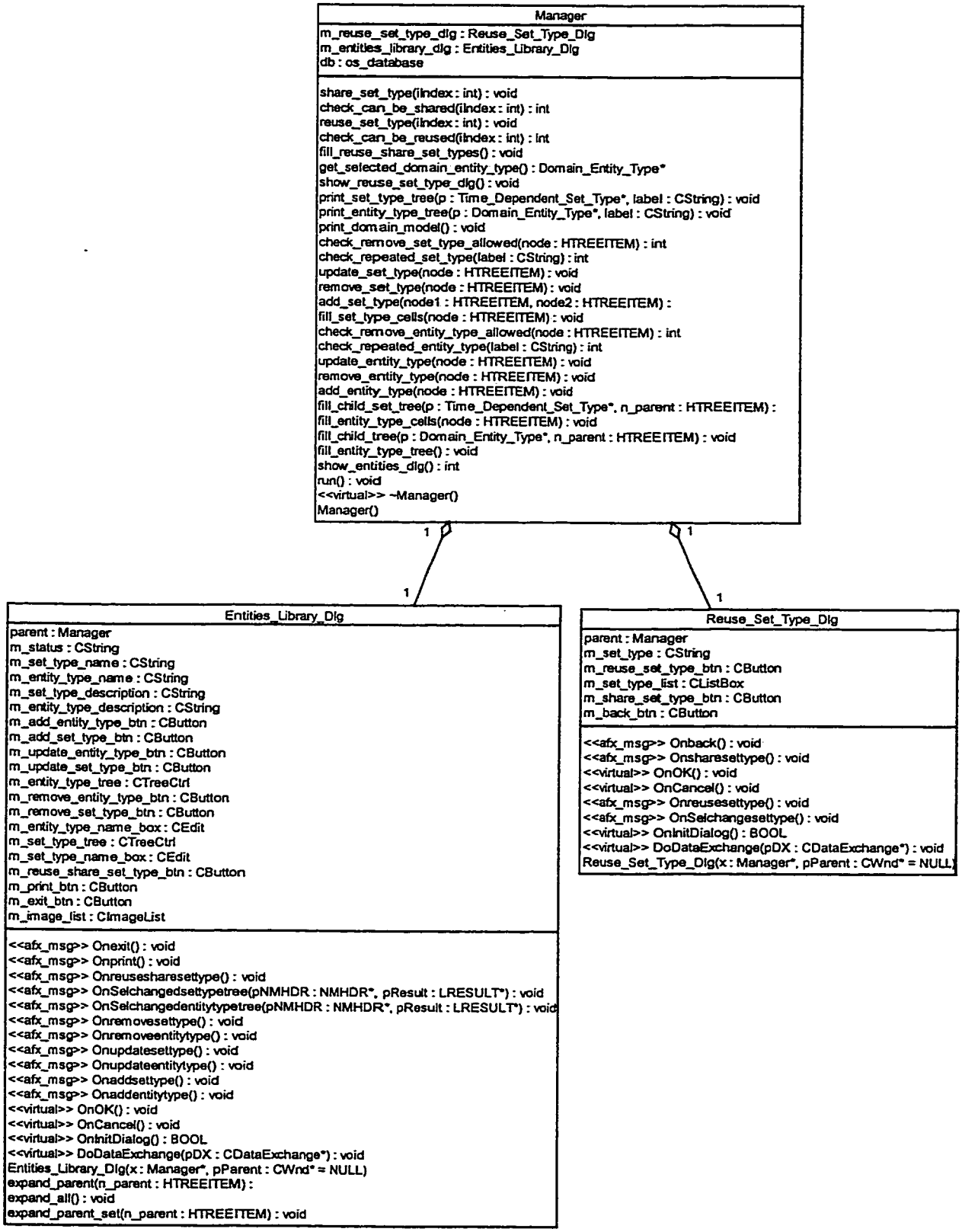
Persistent Part of the Design Model:



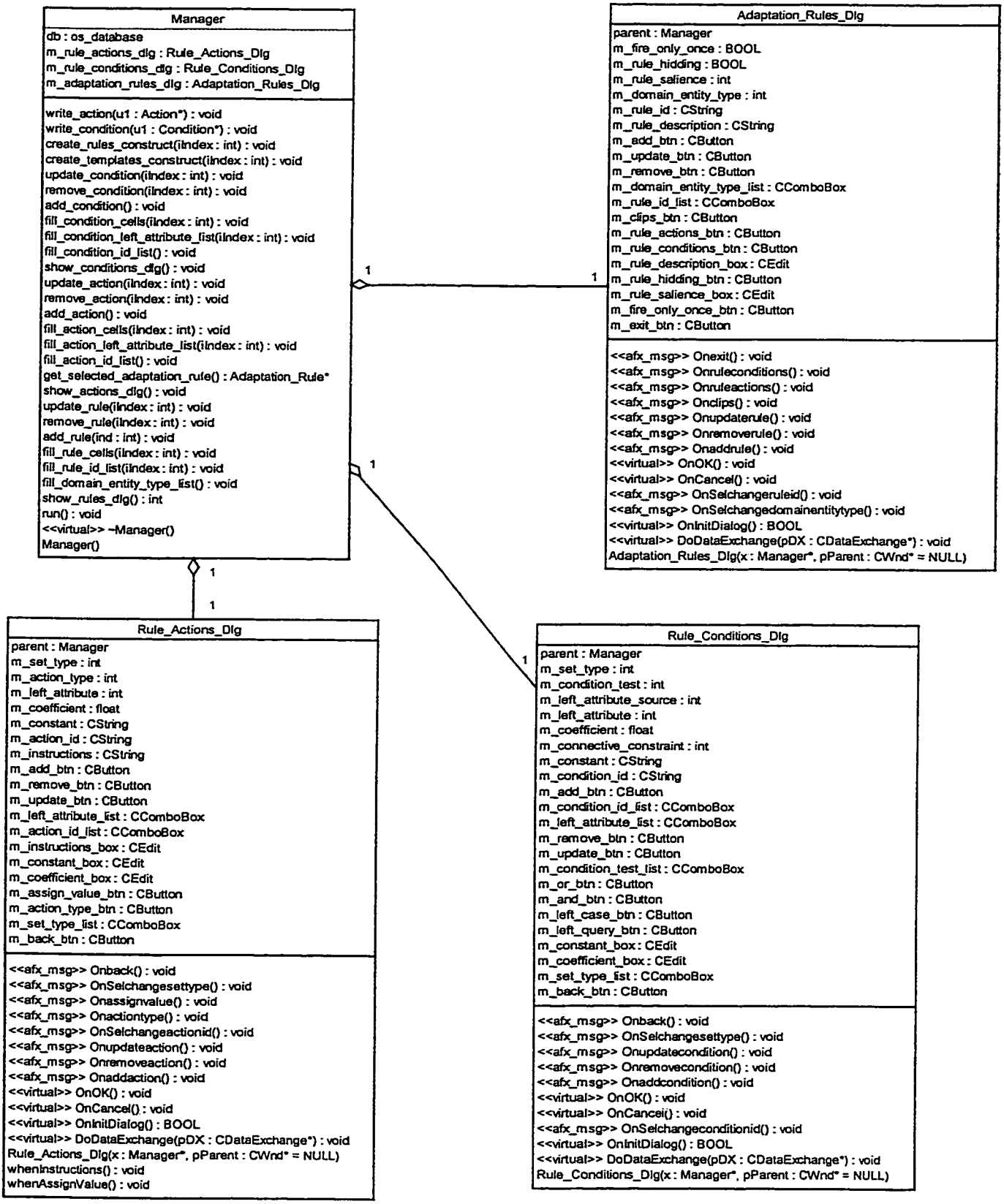
Domain Model Module:



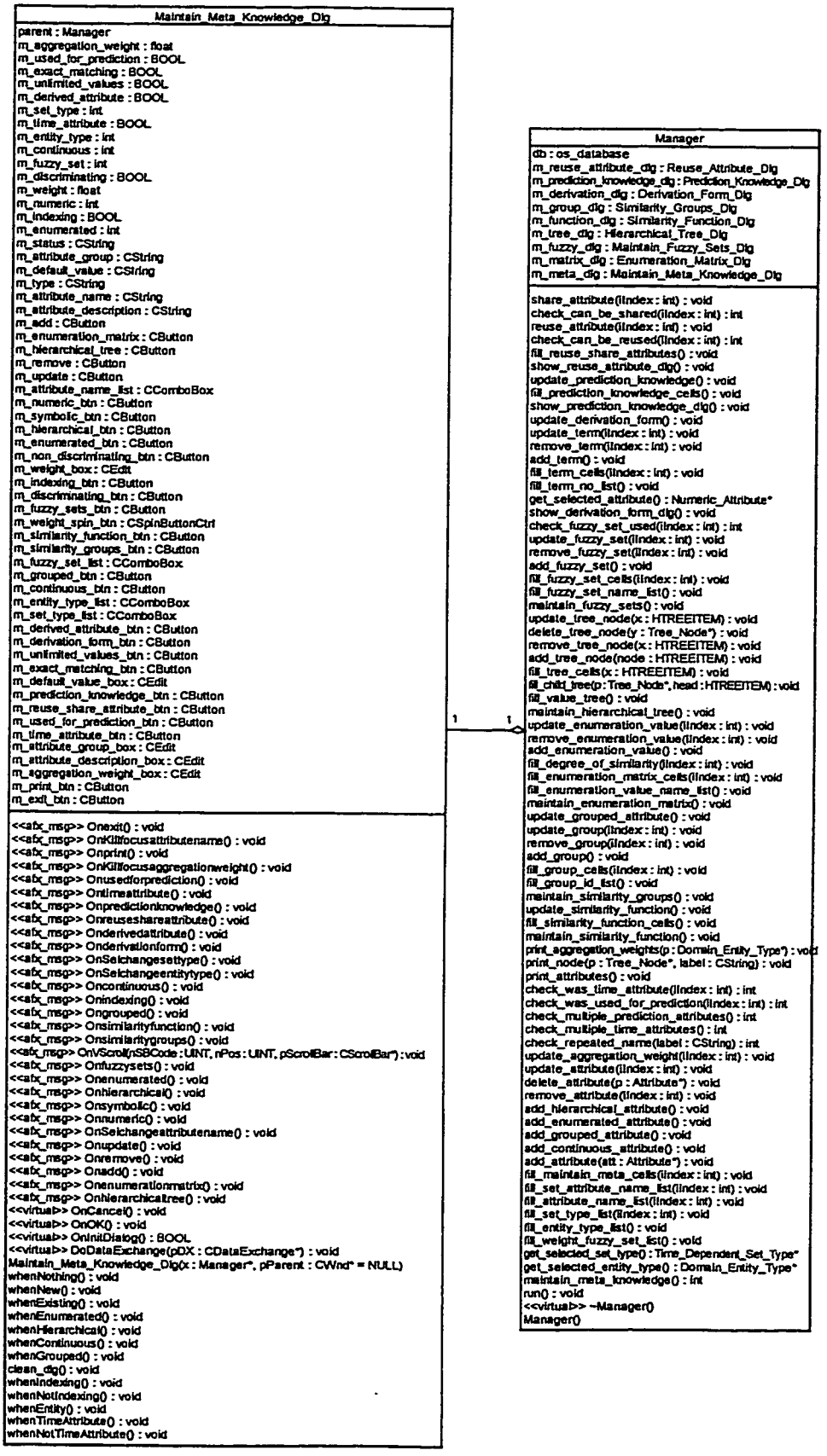
Case Template Library Module:

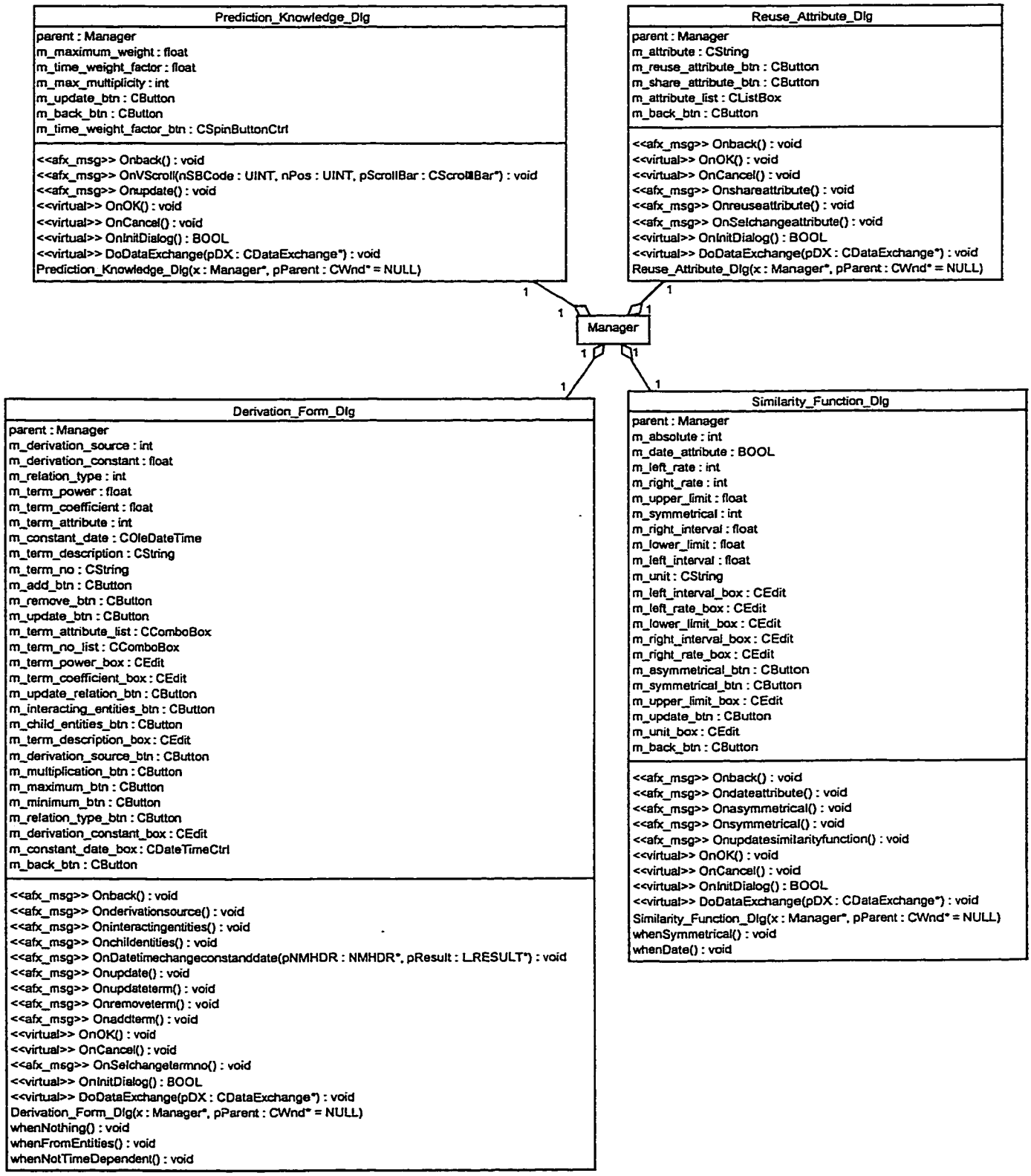


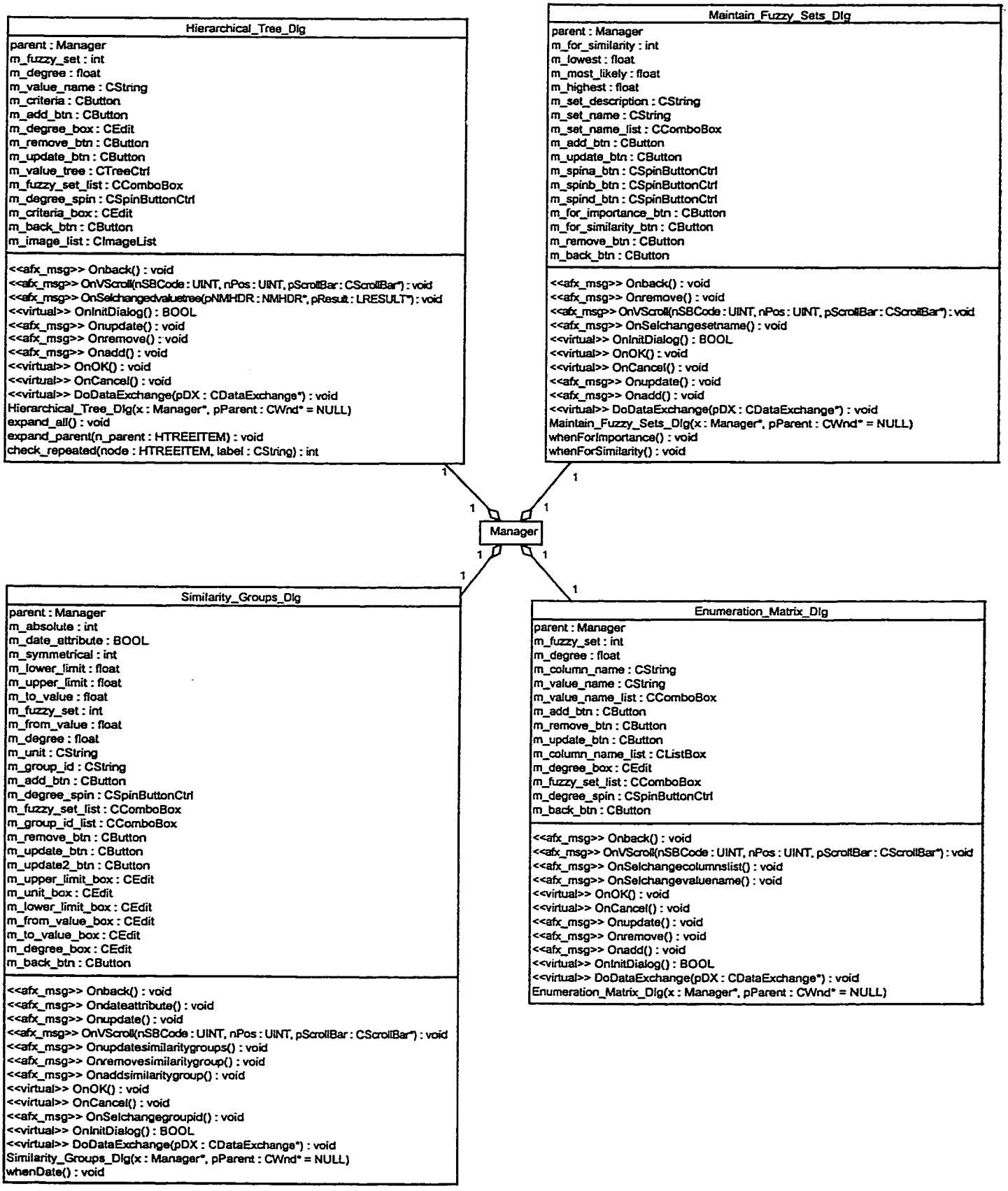
Adaptation Knowledge Module:



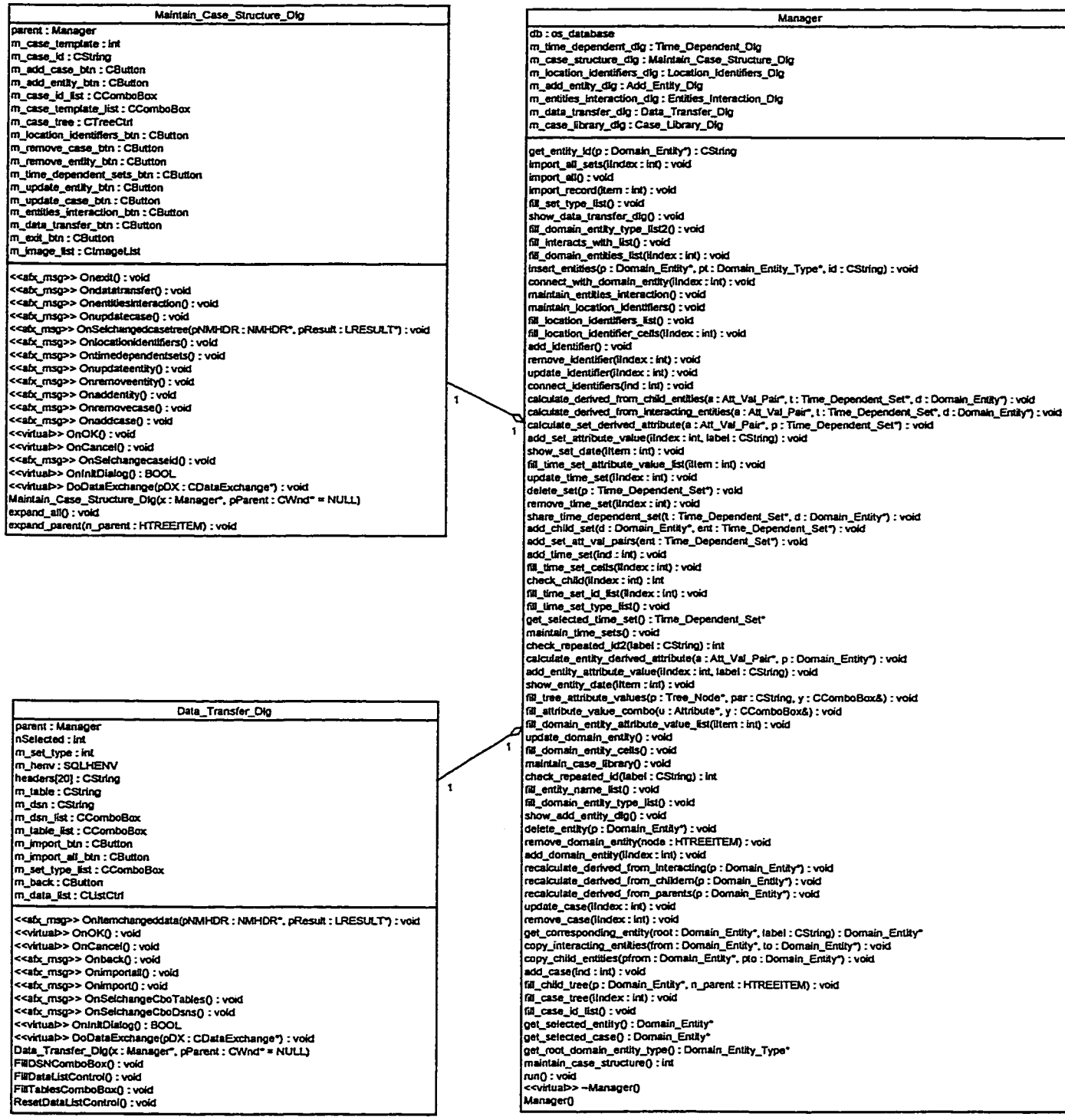
Retrieval Knowledge Module:

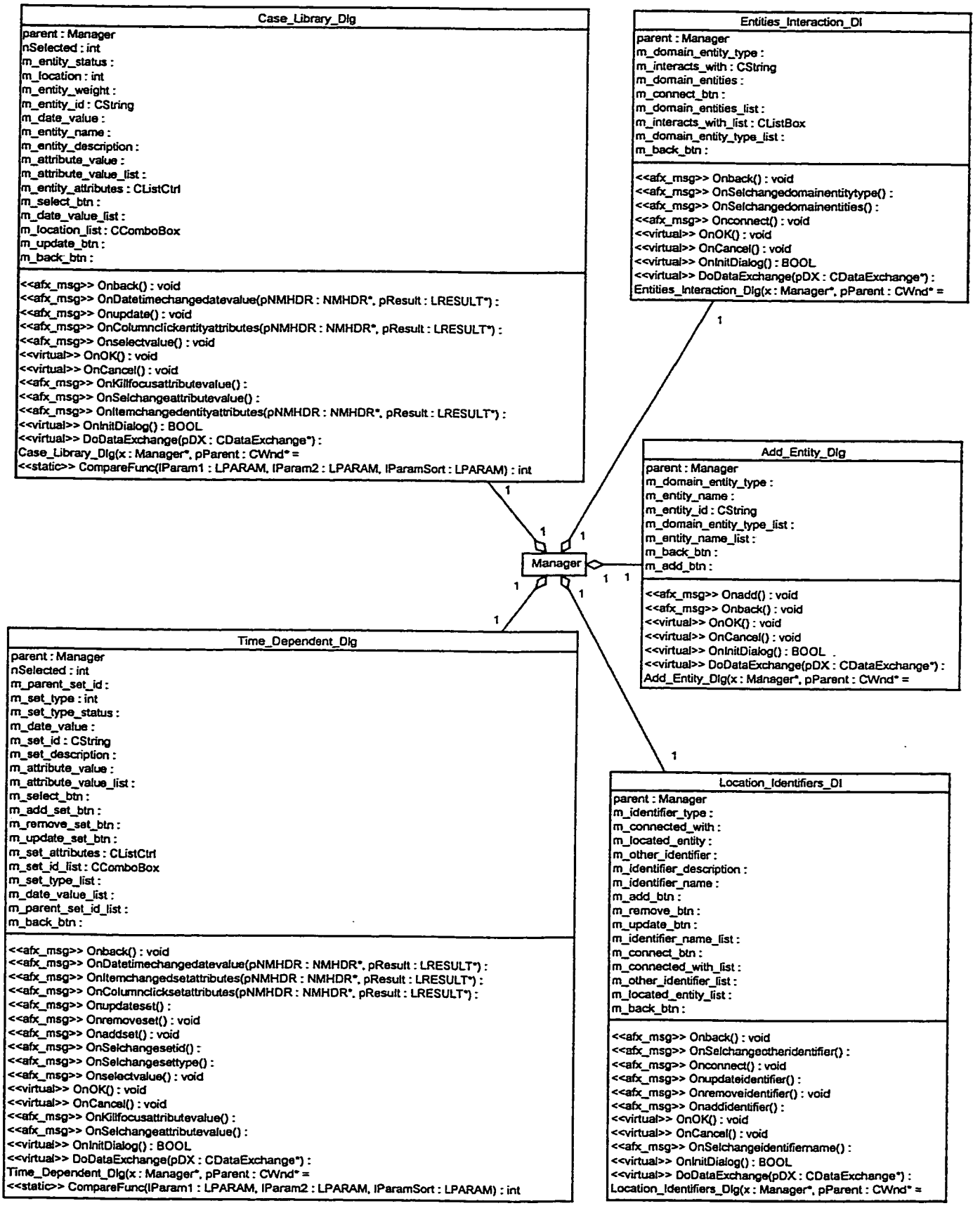



Prediction_Knowledge_Dlg
parent : Manager
m_maximum_weight : float
m_time_weight_factor : float
m_max_multiplicity : int
m_update_btn : CButton
m_back_btn : CButton
m_time_weight_factor_btn : CSpinButtonCtrl
<<afx_msg>> Onback() : void
<<afx_msg>> OnVScroll(nSBCode : UINT, nPos : UINT, pScrollBar : CScrollBar*) : void
<<afx_msg>> Onupdate() : void
<<virtual>> OnOK() : void
<<virtual>> OnCancel() : void
<<virtual>> OnInitDialog() : BOOL
<<virtual>> DoDataExchange(pDX : CDataExchange*) : void
Prediction_Knowledge_Dlg(x : Manager*, pParent : CWnd* = NULL)
Reuse_Attribute_Dlg
parent : Manager
m_attribute : CString
m_reuse_attribute_btn : CButton
m_share_attribute_btn : CButton
m_attribute_list : CListBox
m_back_btn : CButton
<<afx_msg>> Onback() : void
<<virtual>> OnOK() : void
<<virtual>> OnCancel() : void
<<afx_msg>> Onshareattribute() : void
<<afx_msg>> Onreuseattribute() : void
<<afx_msg>> OnSelchangeattribute() : void
<<virtual>> OnInitDialog() : BOOL
<<virtual>> DoDataExchange(pDX : CDataExchange*) : void
Reuse_Attribute_Dlg(x : Manager*, pParent : CWnd* = NULL)
Manager
1
Derivation_Form_Dlg
parent : Manager
m_derivation_source : int
m_derivation_constant : float
m_relation_type : int
m_term_power : float
m_term_coefficient : float
m_term_attribute : int
m_constant_date : COleDateTime
m_term_description : CString
m_term_no : CString
m_add_btn : CButton
m_remove_btn : CButton
m_update_btn : CButton
m_term_attribute_list : CComboBox
m_term_no_list : CComboBox
m_term_power_box : CEdit
m_term_coefficient_box : CEdit
m_update_relation_btn : CButton
m_interacting_entities_btn : CButton
m_child_entities_btn : CButton
m_term_description_box : CEdit
m_derivation_source_btn : CButton
m_multiplication_btn : CButton
m_maximum_btn : CButton
m_minimum_btn : CButton
m_relation_type_btn : CButton
m_derivation_constant_box : CEdit
m_constant_date_box : CDateTimeCtrl
m_back_btn : CButton
<<afx_msg>> Onback() : void
<<afx_msg>> Onderivationsource() : void
<<afx_msg>> Oninteractingentities() : void
<<afx_msg>> Onchildentities() : void
<<afx_msg>> OnDatetimechangeconstanddate(pNMHDR : NMHDR*, pResult : LRESULT*) : void
<<afx_msg>> Onupdate() : void
<<afx_msg>> Onupdateterm() : void
<<afx_msg>> Onremoveterm() : void
<<afx_msg>> Onaddterm() : void
<<virtual>> OnOK() : void
<<virtual>> OnCancel() : void
<<afx_msg>> OnSelchangetermno() : void
<<virtual>> OnInitDialog() : BOOL
<<virtual>> DoDataExchange(pDX : CDataExchange*) : void
Derivation_Form_Dlg(x : Manager*, pParent : CWnd* = NULL)
whenNothing() : void
whenFromEntities() : void
whenNotTimeDependent() : void
Similarity_Function_Dlg
parent : Manager
m_absolute : int
m_date_attribute : BOOL
m_left_rate : int
m_right_rate : int
m_upper_limit : float
m_symmetrical : int
m_right_interval : float
m_lower_limit : float
m_left_interval : float
m_unit : CString
m_left_interval_box : CEdit
m_left_rate_box : CEdit
m_lower_limit_box : CEdit
m_right_interval_box : CEdit
m_right_rate_box : CEdit
m_asymmetrical_btn : CButton
m_symmetrical_btn : CButton
m_upper_limit_box : CEdit
m_update_btn : CButton
m_unit_box : CEdit
m_back_btn : CButton
<<afx_msg>> Onback() : void
<<afx_msg>> Ondateattribute() : void
<<afx_msg>> Onasymmetrical() : void
<<afx_msg>> Onsymmetrical() : void
<<afx_msg>> Onupdatesimilarityfunction() : void
<<virtual>> OnOK() : void
<<virtual>> OnCancel() : void
<<virtual>> OnInitDialog() : BOOL
<<virtual>> DoDataExchange(pDX : CDataExchange*) : void
Similarity_Function_Dlg(x : Manager*, pParent : CWnd* = NULL)
whenSymmetrical() : void
whenDate() : void


Hierarchical_Tree_Dlg
parent : Manager
m_fuzzy_set : int
m_degree : float
m_value_name : CString
m_criteria : CButton
m_add_btn : CButton
m_degree_box : CEdit
m_remove_btn : CButton
m_update_btn : CButton
m_value_tree : CTreeCtrl
m_fuzzy_set_list : CComboBox
m_degree_spin : CSpinButtonCtrl
m_criteria_box : CEdit
m_back_btn : CButton
m_image_list : CImageList
<<afx_msg>> Onback() : void
<<afx_msg>> OnVScroll(nSBCode : UINT, nPos : UINT, pScrollBar : CScrollBar*) : void
<<afx_msg>> OnSelchangedvaluetree(pNMHDR : NMHDR*, pResult : LRESULT*) : void
<<virtual>> OnInitDialog() : BOOL
<<afx_msg>> Onupdate() : void
<<afx_msg>> Onremove() : void
<<afx_msg>> Onadd() : void
<<virtual>> OnOK() : void
<<virtual>> OnCancel() : void
<<virtual>> DoDataExchange(pDX : CDataExchange*) : void
Hierarchical_Tree_Dlg(x : Manager*, pParent : CWnd* = NULL)
expand_all() : void
expand_parent(n_parent : HTREEITEM) : void
check_repeated(node : HTREEITEM, label : CString) : int
Maintain_Fuzzy_Sets_Dlg
parent : Manager
m_for_similarity : int
m_lowest : float
m_most_likely : float
m_highest : float
m_set_description : CString
m_set_name : CString
m_set_name_list : CComboBox
m_add_btn : CButton
m_update_btn : CButton
m_spina_btn : CSpinButtonCtrl
m_spinb_btn : CSpinButtonCtrl
m_spind_btn : CSpinButtonCtrl
m_for_importance_btn : CButton
m_for_similarity_btn : CButton
m_remove_btn : CButton
m_back_btn : CButton
<<afx_msg>> Onback() : void
<<afx_msg>> Onremove() : void
<<afx_msg>> OnVScroll(nSBCode : UINT, nPos : UINT, pScrollBar : CScrollBar*) : void
<<afx_msg>> OnSelchangesetname() : void
<<virtual>> OnInitDialog() : BOOL
<<virtual>> OnOK() : void
<<virtual>> OnCancel() : void
<<afx_msg>> Onupdate() : void
<<afx_msg>> Onadd() : void
<<virtual>> DoDataExchange(pDX : CDataExchange*) : void
Maintain_Fuzzy_Sets_Dlg(x : Manager*, pParent : CWnd* = NULL)
whenForImportance() : void
whenForSimilarity() : void
Manager
Similarity_Groups_Dlg
parent : Manager
m_absolute : int
m_date_attribute : BOOL
m_symmetrical : int
m_lower_limit : float
m_upper_limit : float
m_to_value : float
m_fuzzy_set : int
m_from_value : float
m_degree : float
m_unit : CString
m_group_id : CString
m_add_btn : CButton
m_degree_spin : CSpinButtonCtrl
m_fuzzy_set_list : CComboBox
m_group_id_list : CComboBox
m_remove_btn : CButton
m_update_btn : CButton
m_update2_btn : CButton
m_upper_limit_box : CEdit
m_unit_box : CEdit
m_lower_limit_box : CEdit
m_from_value_box : CEdit
m_to_value_box : CEdit
m_degree_box : CEdit
m_back_btn : CButton
<<afx_msg>> Onback() : void
<<afx_msg>> Ondateattribute() : void
<<afx_msg>> Onupdate() : void
<<afx_msg>> OnVScroll(nSBCode : UINT, nPos : UINT, pScrollBar : CScrollBar*) : void
<<afx_msg>> Onupdatesimilaritygroups() : void
<<afx_msg>> Onremovesimilaritygroup() : void
<<afx_msg>> Onaddsimilaritygroup() : void
<<virtual>> OnOK() : void
<<virtual>> OnCancel() : void
<<afx_msg>> OnSelchangegroupid() : void
<<virtual>> OnInitDialog() : BOOL
<<virtual>> DoDataExchange(pDX : CDataExchange*) : void
Similarity_Groups_Dlg(x : Manager*, pParent : CWnd* = NULL)
whenDate() : void
Enumeration_Matrix_Dlg
parent : Manager
m_fuzzy_set : int
m_degree : float
m_column_name : CString
m_value_name : CString
m_value_name_list : CComboBox
m_add_btn : CButton
m_remove_btn : CButton
m_update_btn : CButton
m_column_name_list : CListBox
m_degree_box : CEdit
m_fuzzy_set_list : CComboBox
m_degree_spin : CSpinButtonCtrl
m_back_btn : CButton
<<afx_msg>> Onback() : void
<<afx_msg>> OnVScroll(nSBCode : UINT, nPos : UINT, pScrollBar : CScrollBar*) : void
<<afx_msg>> OnSelchangecolumnslist() : void
<<afx_msg>> OnSelchangevaluename() : void
<<virtual>> OnOK() : void
<<virtual>> OnCancel() : void
<<afx_msg>> Onupdate() : void
<<afx_msg>> Onremove() : void
<<afx_msg>> Onadd() : void
<<virtual>> OnInitDialog() : BOOL
<<virtual>> DoDataExchange(pDX : CDataExchange*) : void
Enumeration_Matrix_Dlg(x : Manager*, pParent : CWnd* = NULL)
1

Case Library Module:



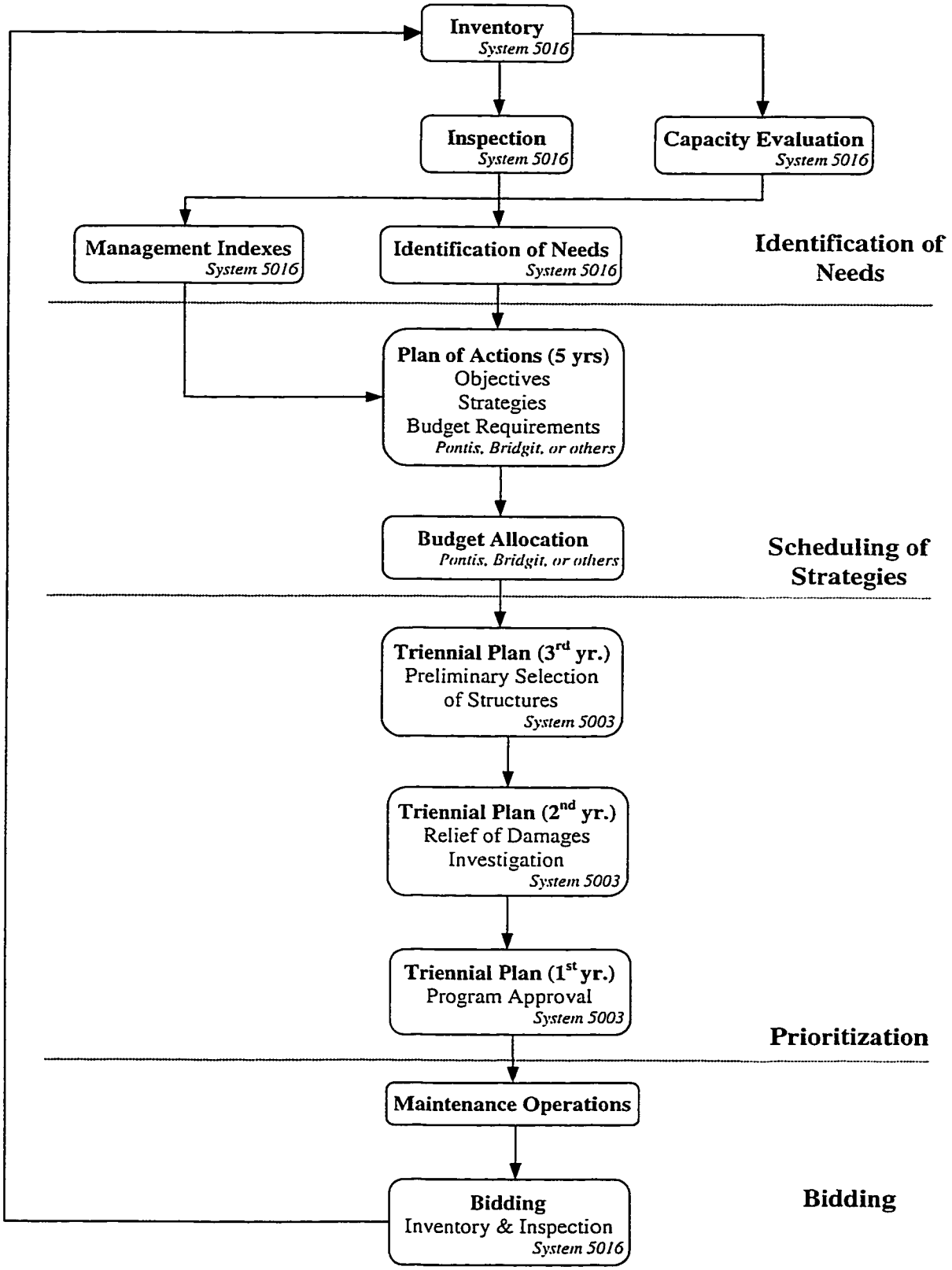

Case_Library_Dlg
parent : Manager
nSelected : int
m_entity_status :
m_location : int
m_entity_weight :
m_entity_id : CString
m_date_value :
m_entity_name :
m_entity_description :
m_attribute_value :
m_attribute_value_list :
m_entity_attributes : CListCtrl
m_select_btn :
m_date_value_list :
m_location_list : CComboBox
m_update_btn :
m_back_btn :
<<afx_msg>> Onback() : void
<<afx_msg>> OnDatetimechangedatevalue(pNMHDR : NMHDR*, pResult : LRESULT*) :
<<afx_msg>> Onupdate() : void
<<afx_msg>> OnColumnclickentityattributes(pNMHDR : NMHDR*, pResult : LRESULT*) :
<<afx_msg>> Onselectvalue() : void
<<virtual>> OnOK() : void
<<virtual>> OnCancel() : void
<<afx_msg>> OnKillfocusattributevalue() :
<<afx_msg>> OnSelchangeattributevalue() :
<<afx_msg>> OnItemchangedentityattributes(pNMHDR : NMHDR*, pResult : LRESULT*) :
<<virtual>> OnInitDialog() : BOOL
<<virtual>> DoDataExchange(pDX : CDataExchange*) :
Case_Library_Dlg(x : Manager*, pParent : CWnd* =
<<static>> CompareFunc(lParam1 : LPARAM, lParam2 : LPARAM, lParamSort : LPARAM) : int
Entities_Interaction_Dl
parent : Manager
m_domain_entity_type :
m_interacts_with : CString
m_domain_entities :
m_connect_btn :
m_domain_entities_list :
m_interacts_with_list : CListBox
m_domain_entity_type_list :
m_back_btn :
<<afx_msg>> Onback() : void
<<afx_msg>> OnSelchangedomainentitytype() :
<<afx_msg>> OnSelchangedomainentities() :
<<afx_msg>> Onconnect() : void
<<virtual>> OnOK() : void
<<virtual>> OnCancel() : void
<<virtual>> OnInitDialog() : BOOL
<<virtual>> DoDataExchange(pDX : CDataExchange*) :
Entities_Interaction_Dlg(x : Manager*, pParent : CWnd* =
Add_Entity_Dlg
parent : Manager
m_domain_entity_type :
m_entity_name :
m_entity_id : CString
m_domain_entity_type_list :
m_entity_name_list :
m_back_btn :
m_add_btn :
<<afx_msg>> Onadd() : void
<<afx_msg>> Onback() : void
<<virtual>> OnOK() : void
<<virtual>> OnCancel() : void
<<virtual>> OnInitDialog() : BOOL
<<virtual>> DoDataExchange(pDX : CDataExchange*) :
Add_Entity_Dlg(x : Manager*, pParent : CWnd* =
Manager
Time_Dependent_Dlg
parent : Manager
nSelected : int
m_parent_set_id :
m_set_type : int
m_set_type_status :
m_date_value :
m_set_id : CString
m_set_description :
m_attribute_value :
m_attribute_value_list :
m_select_btn :
m_add_set_btn :
m_remove_set_btn :
m_update_set_btn :
m_set_attributes : CListCtrl
m_set_id_list : CComboBox
m_set_type_list :
m_date_value_list :
m_parent_set_id_list :
m_back_btn :
<<afx_msg>> Onback() : void
<<afx_msg>> OnDatetimechangedatevalue(pNMHDR : NMHDR*, pResult : LRESULT*) :
<<afx_msg>> OnItemchangedsetattributes(pNMHDR : NMHDR*, pResult : LRESULT*) :
<<afx_msg>> OnColumnclicksetattributes(pNMHDR : NMHDR*, pResult : LRESULT*) :
<<afx_msg>> Onupdateset() :
<<afx_msg>> Onremoveset() : void
<<afx_msg>> Onaddset() : void
<<afx_msg>> OnSelchangesetid() :
<<afx_msg>> OnSelchangesettype() :
<<afx_msg>> Onselectvalue() : void
<<virtual>> OnOK() : void
<<virtual>> OnCancel() : void
<<afx_msg>> OnKillfocusattributevalue() :
<<afx_msg>> OnSelchangeattributevalue() :
<<virtual>> OnInitDialog() : BOOL
<<virtual>> DoDataExchange(pDX : CDataExchange*) :
Time_Dependent_Dlg(x : Manager*, pParent : CWnd* =
<<static>> CompareFunc(lParam1 : LPARAM, lParam2 : LPARAM, lParamSort : LPARAM) : int
Location_Identifiers_Dl
parent : Manager
m_identifier_type :
m_connected_with :
m_located_entity :
m_other_identifier :
m_identifier_description :
m_identifier_name :
m_add_btn :
m_remove_btn :
m_update_btn :
m_identifier_name_list :
m_connect_btn :
m_connected_with_list :
m_other_identifier_list :
m_located_entity_list :
m_back_btn :
<<afx_msg>> Onback() : void
<<afx_msg>> OnSelchangeotheridentifier() :
<<afx_msg>> Onconnect() : void
<<afx_msg>> Onupdateidentifier() :
<<afx_msg>> Onremoveidentifier() : void
<<afx_msg>> Onaddidentifier() :
<<virtual>> OnOK() : void
<<virtual>> OnCancel() : void
<<afx_msg>> OnSelchangeidentifiername() :
<<virtual>> OnInitDialog() : BOOL
<<virtual>> DoDataExchange(pDX : CDataExchange*) :
Location_Identifiers_Dlg(x : Manager*, pParent : CWnd* =

Case Retrieval and Adaptation Module:

APPENDIX E
Structure Management System in Quebec

List of Items:

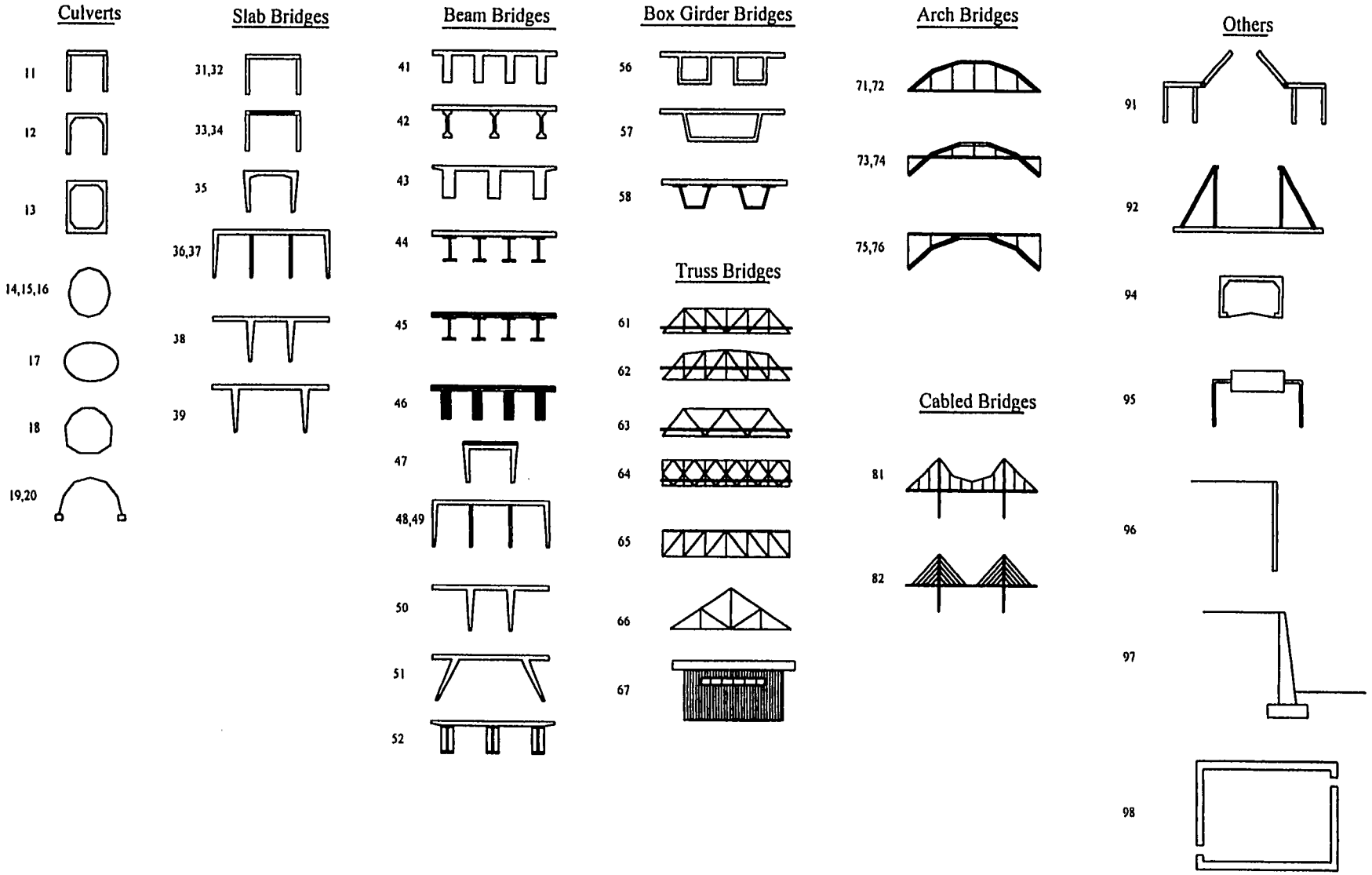
Organization of MTQ Structure Management System (MTQ 1997)



Distribution of structure types in Quebec

Category	ID	Material	System	Frequency	Total	
Culvert					881	9.3%
	11	R.C.	Solid slab	14		
	12	R.C.	Rigid frame	0		
	13	R.C.	Box section	392		
	14	R.C.	Circular section	1		
	15	Steel	Circular section	94		
	16	Thermoplastic	Circular section	0		
	17	Steel	Elliptic section	19		
	18	Steel	Curved closed section	244		
	19	R.C.	Arc	62		
	20	Steel	Arc	55		
Slab Bridge					1749	18.4%
	31	R.C.	Solid slab	575		
	32	P.C.	Solid slab	36		
	33	R.C.	Hollow slab	151		
	34	P.C.	Hollow thick slab	22		
	35	R.C.	Portal frame	445		
	36	R.C.	Portal frame below ground	353		
	37	P.C.	Portal frame	0		
	38	R.C.	Rigid frame	157		
	39	P.C.	Rigid frame	10		
Beam Bridge					5679	59.8%
	41	R.C.	Rectangular beams	1483		
	42	P.C.	Precast beams	722		
	43	P.C.	Rectangular beams	177		
	44	Steel	I-beams under R.C. slab	644		
	45	R.C.	I-beams under wood slab	2363		
	46	Wood	Rectangular beams	30		
	47	R.C.	Portal frame	28		
	48	R.C.	Portal frame below ground	1		
	49	Steel	Portal frame	0		
	50	R.C.	Rigid frame	50		
	51	Steel	Rigid frame	8		
	52	Steel	Covered with concrete	173		
Box-Girder Bridge					139	1.5%
	56	R.C.	Two boxes	55		
	57	P.C.	One box	48		
	58	Steel	Two boxes	36		
Truss Bridge					305	3.2%
	61	Steel	Through N truss	97		
	62	Steel	Intermediate N truss	2		
	63	Steel	Through W truss	78		
	64	Steel	Through bailey truss	12		
	65	Steel	Deck N truss	32		
	66	Wood	Triangular truss	4		
	67	Steel	Covered truss	80		
Arc Bridge					73	0.8%
	71	R.C.	Though arch	4		
	72	Steel	Though arch	12		
	73	R.C.	Intermediate arch	1		
	74	Steel	Intermediate arch	0		
	75	R.C.	Deck arch	51		
	76	Steel	Deck arch	5		
Cabled bridge					10	0.1%
	81	Any	Suspension bridge	5		
	82	Any	Cable-Stayed bridge	5		
Others					664	7.0%
	91	Any	Movable bridge	1		
	92	Any	Foot bridge	36		
	94	Any	Tunnel	36		
	95	Any	Signals support	0		
	96	Any	Platform	0		
	97	Any	Retaining wall	566		
	98	Any	Pumping station	24		
	99	Any	Others	1		
				Total Number of Structures	*9500*	100%

Figures of structure types in Quebec (MTQ 1997)



Inspection forms used in Quebec (MTQ 1995)

Inspection Form	Form Description	No. of Inspected Elements
A	Signalisation	4
B	Water stream, approaches backfilling, and slope protection	5
C	Abutment and wing wall	7
D	Foundation and substructure	6
E	Deck	7
F	Solid beam	7
G	Box girder	8
H	Truss beam	5
I	Arc beam	6
J	Arc with spanderal wall	9
K	Structure floor	7
L	Wind bracings	5
M	Deck joints	7
N	Curbs and side walks	3
O	Barriers	9
P	Approaches	7
Q	Retaining walls	6
R	Culverts	6
S	Covered bridge	11
U	Suspension bridge	7
V	Cable-Stayed bridge	3
		135

Example of Inspection Form E (MTQ 1995)

Identification of concerned spans:

No.	Type	Element	MCR	PCR	Description
1	S	Wearing surface			
2	A	Drainage system			
3	P	Exterior face ( )			
4	P	Exterior face ( )			
5	P	End soffit ( )			
6	P	Middle soffit ( )			
7	P	End soffit ( )			

Remarks

Recommendations

Activity	Description	Quantity	Unit

Balance factors used in Quebec (MTQ 1995)

Inspection Form	Element Number										
	1	2	3	4	5	6	7	8	9	10	11
C	0.500	0.150	0.100	0.100	0.050	0.050	0.050				
D	0.500	0.150	0.100	0.050	0.100	0.100					
E	0.000	0.000	0.200	0.200	0.200	0.200	0.200				
F	0.220	0.220	0.220	0.060	0.110	0.060	0.110				
G	0.150	0.150	0.150	0.080	0.080	0.150	0.090	0.150			
H	0.350	0.250	0.150	0.150	0.100						
I	0.250	0.180	0.180	0.130	0.130	0.130					
J	0.125	0.500	0.150	0.150	0.150	0.150	0.150	0.500	0.125		
K	0.100	0.300	0.100	0.113	0.225	0.113	0.050				
L	0.125	0.125	0.250	0.250	0.250						
N	0.350	0.300	0.350								
Q	1.000	1.000	0.000	0.000	0.000	0.000					
R	0.250	0.250	0.250	0.500	0.500	0.250					
S	0.100	0.150	0.100	0.100	0.100	0.150	0.100	0.100	0.050	0.050	0.000
U	0.150	0.250	0.250	0.250	0.050	0.050	0.000				
V	0.500	0.300	0.200								