CRC Construction Innovation
BUILDING OUR FUTURE

Case-Based Reasoning in Construction and Infrastructure Projects - Final Report

Report No. 2002-059-B No.16

Editor Penny Corrigan

The research described in this report was carried out by

Project Leader Ivan Cole

Team Members:

Michael Ball
Alan Carse
Wan Yee Chan
Penny Corrigan
Wayne Ganther
Tim Muster
David Paterson
Gerardo Trinidad
Mary Lou Maher
PakSan Liew

Research Program: B

Program Name Sustainable Built Assets

Project No.: 2002-059-B

Project Name: Case-Based Reasoning in Construction and Infrastructure Projects

Date: June 2005

Distribution List

Cooperative Research Centre for Construction Innovation
Authors:
Ivan Cole
Michael Ball
Alan Carse
Wan Yee Chan
Penny Corriga
Wayne Ganther
Tim Muster
David Paterson
Gerardo Trinidad
Mary Lou Maher
PakSan Liew

Disclaimer

The Client makes use of this Report or any information provided by the Cooperative Research Centre for **Construction Innovation** in relation to the Consultancy Services at its own risk. Construction Innovation will not be responsible for the results of any actions taken by the Client or third parties on the basis of the information in this Report or other information provided by Construction Innovation nor for any errors or omissions that may be contained in this Report. Construction Innovation expressly disclaims any liability or responsibility to any person in respect of any thing done or omitted to be done by any person in reliance on this Report or any information provided.

© 2005 Icon.Net Pty Ltd

To the extent permitted by law, all rights are reserved and no part of this publication covered by copyright may be reproduced or copied in any form or by any means except with the written permission of Icon.Net Pty Ltd.

Please direct all enquiries to:

Chief Executive Officer
Cooperative Research Centre for Construction Innovation
9th Floor, L Block, QUT, 2 George St
Brisbane Qld 4000
AUSTRALIA
T: 61 7 3864 1393
F: 61 7 3864 9151
E: enquiries@construction-innovation.info
W: www.construction-innovation.info

Table of Contents

List of Tables

List of Figures

EXECUTIVE SUMMARY

The project has applied the concept of case-based reasoning to prediction of lifetime for metallic building components. This was considered a suitable method to combine a range of different data sources and determine the most appropriate answer for any given situation.

Discussions with the project partners identified two areas of particular interest for formulation of initial applications.

The project has delivered:

- Design and implementation of a case-based reasoning (CBR) engine for life prediction of metallic building components in general,
- An application of the CBR engine tailored to predicting durability of gutters in Queensland schools,
- A stand-alone program for modelling the degradation rate of gutters using the CSIRO holistic model,
- A stand-alone program for estimating salt deposition levels on bridge structures in Queensland to be used as the basis for a CBR program in the future, and
- A report on the Sunshine Coast site visit to school and bridge locations, which has identified several corrosion problems of interest to the industry partners.

The implementation of the CBR engine necessitated characterisation of the environment and building locations to enable development of case definitions. Similarity rules were formulated for a number of parameters so that different cases could be compared and the closest match selected.

The QDPW application incorporates several sources of data for access by the CBR engine. These include the Delphi survey (from Project 2002-010-B), maintenance information from the QDPW and the holistic model. The holistic model required modifications to tailor the outputs for use with gutters. The three main materials currently used in gutters are galvanised steel, Zincalume and Colorbond® so rules for the degradation of polymeric coatings had to be determined and included in the model for use with Colorbond®. In addition, experiments were carried out to determine an appropriate 'Time of Wetness' factor for different gutter states, given that they are a building component where dirt can accumulate and affect the run-off of water and drying rate. The modelling calculations result in a mass loss per year for metals so this had to be related to a predicted life span, with consideration also given to whether this is aesthetic life or service life. These modifications to the holistic model were incorporated into a stand-alone program which can be used to estimate degradation of gutters at any location in Australia.

The QDMR application is not as advanced as the gutter application. The project team has focussed on the definition of structural elements of five typical Queensland bridges to define representative cases which could be used in a future extension into CBR. A detailed CFD analysis of salt deposition on the five bridge structures has been carried out and elements with common deposition rates were identified. A stand-alone program has been developed that will estimate a salt deposition factor for a selected bridge element at any location in Queensland.

The development of these applications will provide economic benefits to the two industry partners. These are difficult to quantify but contain elements of design savings and maintenance savings for facility owners, managers and maintenance providers. The potential for the tools is significant given the amount of metal used in the areas of interest and the levels of corrosion found in the project site visit to the Sunshine Coast. It has been estimated that nearly $5 million was spent by Queensland Department of Public Works in 03/04 in replacing corroded metallic components of Queensland schools. Substantial cost savings can be made through the use of the S/W tool to select construction materials suited to the environment in which they will be used, and optimisation of maintenance schedules.

1. INTRODUCTION

1.1 Background

Many processes in design, construction and maintenance of infrastructure are complex and highly influenced by a wide range of design, climate and usage parameters. For example predicting corrosion rates, and hence component life, is a complex process which includes reasoning about examples in which corrosion rates are known, knowledge of the material properties and the impact of the environment on those materials, and an interpretation of the site.

The ability to accurately predict the lifetime of building components is crucial to optimising building design, material selection and scheduling of required maintenance. ISO 15686 (Clause 9) has suggested the factor method as a means of estimating the service life of a particular component or assembly in a specific set of conditions. The factor method is based on a reference service life (RSL), which is defined as the expected service life of a component or assembly situated in a well-defined set of conditions. It incorporates a series of modifying factors that relate to the specific conditions of the case to give the predicted service life distribution of a component (PSDLC) according to the equation:

$$PSLDC = RSLC \cdot f_A \cdot f_B \cdot f_C \cdot f_D \cdot f_E \cdot f_F \cdot f_G \quad \text{…Eqn 1.1}$$

The factor indices relate to quality of the component, design, work execution, environments etc. The problem still remains, however, of defining the reference service life for a vast array of building components.

Two approaches have been used in the past to predict the corrosion process - statistical and process-based models. Statistical models have proven unable to cope with the complexity of the problem. Studies have demonstrated that statistical models of component life though useful are extremely limited in their application and cannot predict outside the data sets used to generate the models. Thus a statistical model of life of reinforced concrete in bridges in inland NSW is unlikely to be useful for predicting life for the same bridges on the coast and could not predict life of reinforced concrete in buildings etc.

Process-based models are much more flexible, for example the Construction Mapping System (CMS) developed by CSIRO can predict the life of galvanised steel within any building anywhere in the country. This method is based on the holistic model, within which processes controlling corrosion across a wide range of physical scales and based on different phenomena are modelled. A solution to component life prediction is generated by post-processing the corrosion rate obtained from combining different modules defining specific processes through first principles. Although the theoretical component life of a component can be calculated for any applicable area within Australia, the accuracy of this result reduces dramatically when input data crosses the boundary conditions of the model.

1.2 Problem Definition

The problem is to combine the two approaches to corrosion prediction so that a variety of sources of data, from studies, from experience and from first principles using the holistic model, can be combined to form the basis of the lifetime prediction tool. In addition, once the predicted lifetime for a particular situation has been determined, then this should be available for future reference. Thus, the required system must be able to store, manipulate and compare numerous use-case scenarios. Case-based reasoning is seen as an ideal method for linking together the different data sources and reusing previous experiences in the current context to solve new problems.

The Queensland Department of Public Works and Queensland Department of Main Roads require a means to predict the life of building components subjected to atmospheric corrosion. This tool will form the basis for maintenance optimisation and risk assessment used in developing asset replacement and repair strategies. In particular the software tools to be developed by this project will provide information on life prediction of gutters (as used in Queensland state schools) and life prediction of metal components in Queensland bridges.

The development of these applications will provide economic benefits to the two industry partners. These are difficult to quantify but contain elements of design savings and maintenance savings for facility owners, managers and maintenance providers. The potential for the tools is significant given the amount of metal used in the areas of interest and the levels of corrosion found in a site visit carried out in September 2004 as part of the project. It has been estimated that nearly $5 million was spent by Queensland Department of Public Works in 03/04 in replacing corroded metallic components in Queensland schools. Substantial cost savings can be made through the use of the project software tool to select construction materials suited to the environment in which they will be used.

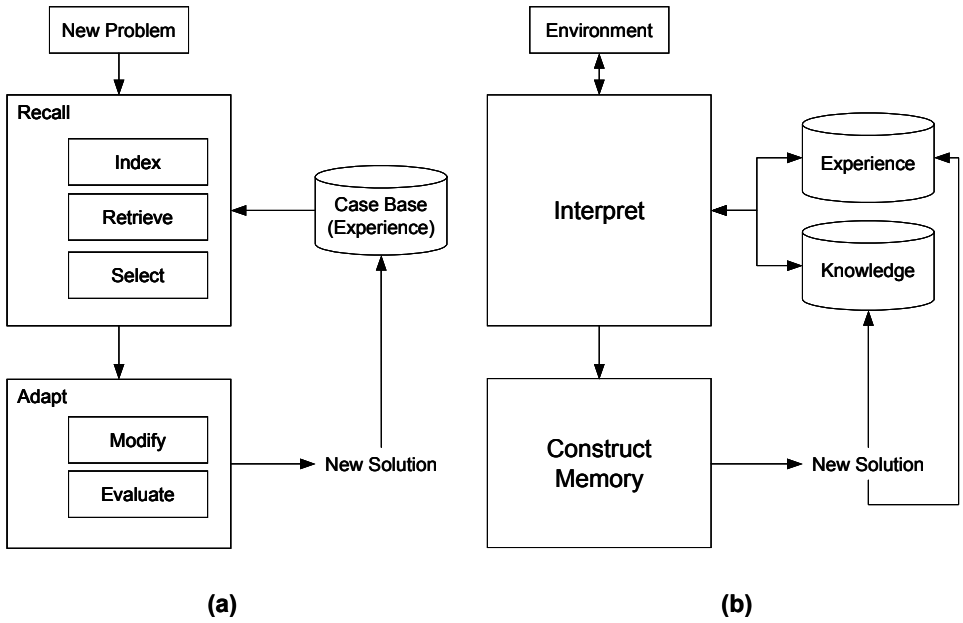
The development of these two applications is discussed in this report, starting with a discussion of the benefits of situated case-based reasoning in Chapter 2. Chapter 3 looks at how this can be applied to the prediction of corrosion rates for metal building components in general. This is followed by the documentation of the design and implementation of the CBR engine in Chapter 4. The development of the two specific applications for the CRC industry partners are discussed in Chapters 5 and 6. The site visit to selected schools and bridges on the Sunshine Coast is summarised in Chapter 7 and suggestions for future extensions of the work are addressed in Chapter 8.

2. SITUATED CASE-BASED REASONING MODEL

2.1 Definition

Case –based reasoning (CBR) provides a model for design reasoning based on the use of a set of previous design experiences represented as design cases (Maher et al 1995). These cases are indexed and retrieved using information about a current design problem, and then through analogical reasoning, a selected case (or set of cases) is adapted until it satisfies the current design specifications and constraints. One aspect of design reasoning that is not addressed by traditional models for case-based reasoning is that designing is situated (Gero 1998). To accommodate the notion of situatedness in designing, the basic idea of case-based reasoning is extended to create a model of situated case-based reasoning (situated CBR) Figure 2.1, based on a model of constructive memory that operates within a framework of situatedness.

Figure 2.1 A conventional case-based reasoning model (a) and a situation case-based reasoning model (b)



In the situated CBR model, instead of focusing on just the design problem and finding a solution to it, emphasis is also given to the environment within which the problem is framed. The model interprets the environment according to the current situation and the problem is framed accordingly. This interpretation is dependent on the current environment, the internal state of the situated CBR system and the interactions between the system and the environment.

The internal state of a situated CBR system is defined by its content. This content is made up of individual entities that are classified either as experience or knowledge. Interactions between the system and the environment define different interpretations of the environment according to different interpretations of the selected entities used for memory construction.

A distinctive characteristic of situated CBR is the way the knowledge and experience are understood and used. In CBR, retrieved cases provide a solution or a starting point for case

adaptation. In Situated CBR, the memory of an experience and/or knowledge (entities) is constructed according to an interpretation of the environment and an interpretation of the selected entities relevant to the problem at hand. Rather than adapt a selected case to new design specifications, the selected entities are interpreted according to the interactions between the system and the environment. These interactions provide a specific view (interpretation) of the relationship between the design specifications and the environment. This view dictates another interpretation of the environment that can introduce new specifications. This "feedback" loop causes the interpretations of the environment and the selection of experiences and knowledge to occur recursively until a common interpretation is reached.

The recursive interpretations of the environment and the selected entities result in new memories as well as new indices to the selected experiences and knowledge to be created. Memories are constructed by:

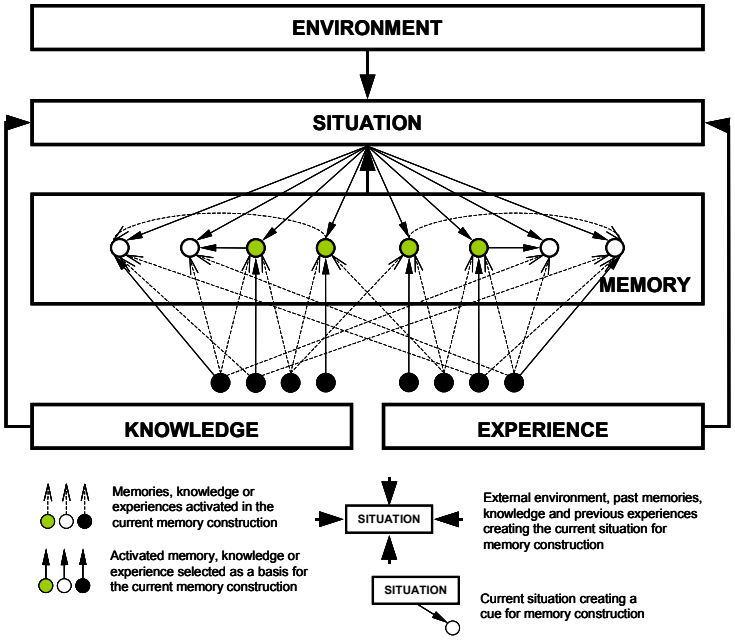
- instantiating the parameter values of the selected entities according to the current situation;
- mapping existing parameters in the selected entities to new ones through an analogical process; and
- restructuring the selected entities according to the current situation.

This is similar to creation of new functional or behavioural indices to an old design prototype within the domain of situated analogy (Gero and Kulinski, 2000).

2.2 Constructive Memory Model

Figure 2.2 illustrates a conceptual model of memory construction. Memories are constructed according to the environment, the knowledge and experience of a situated computational system and the interactions between the system and the environment.

Figure 2.2 Conceptual model of memory construction (from Gero 1999)



Knowledge can be considered as general facts that can refer to a generalised or compiled construct. (Rosenman et al. 1991) such as a design prototype (Gero 1990) that collects function, behaviour and structure information related to designing within a single representation. Methods to acquire knowledge include:

- Abstraction over classes of objects, as in the case of design prototypes,
- Generalisation over facts as in the case of design guidelines (Boothroyd 1994), rules (Witten and Frank 2000) or formulas; or
- Direct learning from external sources such as books, domain experts.

The generic nature of knowledge implies that it does not carry with it any particular solution. A particular situation has to be fitted to the knowledge and a solution has to be inferred from it.

Experience refers to previous episodes recorded in or encountered by the system. It entails the system's involvement as the "first person" in dealing with the substance of that episode. This form of experience can refer to experimental data collected under controlled conditions or to information obtained by data logs.

Memory is a construct created "here and now" for the purpose of operating within the current environment according to a design goal. Knowledge and experience provide the base for constructing a memory according to the current situation.

Memory construction commences with the current situation providing the cue to start off the process. Related knowledge and experience are activated according to the cue and the relevant knowledge and experience are selected as a basis for memory construction. After this selection, the environment and the selected experience are recursively interpreted to construct a memory. This memory contains the required actions to be effected to the environment according to the current design goal. Each memory, after it has been constructed, is added to the system as a new experience by augmenting its representation experience. This new experience is available for subsequent memory constructions.

2.3 Framework of Situatedness

The model of constructive memory resides within the framework of situatedness in designing (Liew and Gero 2004). This notion of situatedness encompasses the fundamental ideas of interaction, memory construction and interpretation.

Interaction implies that the content of a situated CBR system are not encoded a priori and indexed for use later, but rather the content of the system is developed through interactions with the environment. The development of this content entails the construction of a memory about related entities, influenced by any knowledge and experience gained since the entities, and interpreted by the prevailing situation.

Memory construction provides the basis for the recursive interpretation of the current situation. This process of constructing new memory is similar to the notion of "re-representation" described in Gero and Kulinski 2000. A constructed memory defines both the interpretation of the relevant content and the interpretation of the environment in light of the current interactions between the system and the environment. The content of the situated CBR system is interpreted through a "filter" defined by the present situation. An

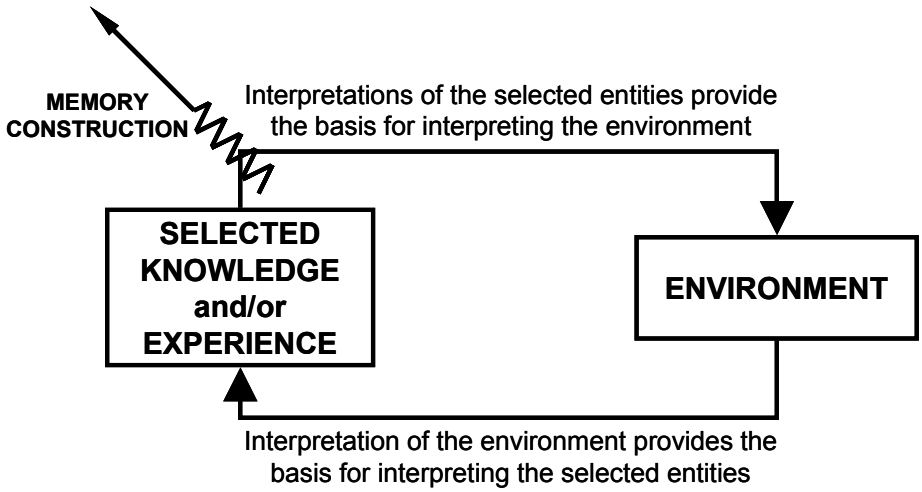
experience and/or knowledge is not “copied” into the present but rather it is interpreted according to the current situation. This interpretation situates the relevant content of the system through the current environment so that it is not necessary to encode all possible forms of know-how a priori. New interpretations of past knowledge and experience is produced by every constructed memory of these entities. This new interpretation is added to the memory system as a new experience and is interpreted later as if it were part of the original content of the system.

A constructed memory also interprets the environment according to the current expectation of the system. This expectation is derived from the goal of the system captured within the relevant entities selected for memory construction. The expectation dictates what is to be focused upon, and the way the environment is to be interpreted in order for the system to perform its task according to its goal.

2.3.1 Recursive Interpretations

Figure 2.3 illustrates the recursive interpretations between the environment and the selected experience and/or knowledge used for memory construction. Both the environment and the selected entities are interpreted through the lens of the current interaction in this recursive fashion. The recursion behaviour is resolved when the interpretations of the environment or selected entities remain the same after a complete cycle of interpretations: interpretation of the environment or selected entities followed by an interpretation of the selected entities or environment. The final constructed memory provides a coherent interpretation of the environment and of relevant knowledge and experience within a single cohesive structure.

Figure 2.3 Recursive interpretations of a selected memory, experience or knowledge and the environment



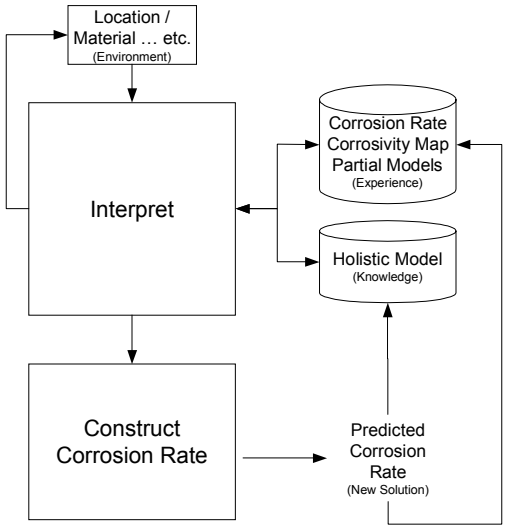
After the successful interpretation of the environment and selected entity, the actions that were performed previously are transformed into a suitable form for the current situation and effected into the environment, as external actions, or to the system, as internal actions, to perform the necessary task according to the current goal of the system.

2.4 Using Situated CBR for Prediction of Corrosion Rates

The situated CBR model will be used to design a system that predicts the atmospheric corrosion of building materials as shown in Figure 2.4. Predicting corrosion rates is a complex process which includes reasoning about examples in which corrosion rates are known, knowledge of the material properties and the impact of the environment on those materials, and an interpretation of the site in which the material is located.

The local conditions of the site in which the material is located are used to determine the environmental component of the situated CBR system. Parameters within this environment are used to select previous experiences and/or knowledge from the system for memory construction. A memory is constructed based on a combination and interpretation of previous experiences that can be used to predict the atmospheric corrosion rate of a specific material on a specific site.

Figure 2.4 Model of situated CBR applied to the prediction of atmospheric corrosion rate

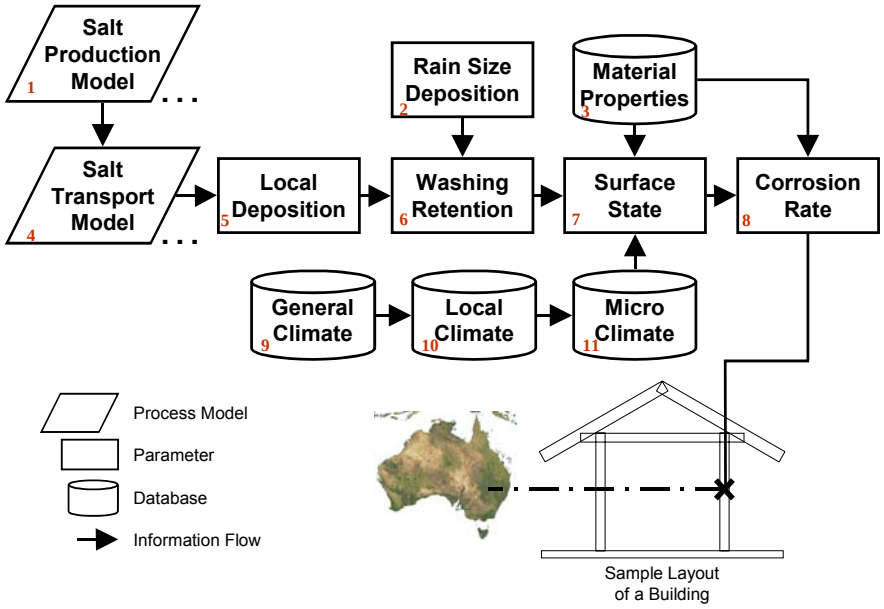


A selected experience may be relevant if a corrosion rate has already been calculated in a similar situation. The differences in conditions are examined to see if they are significant to warrant additional interpretation and computation. If a selected previous experience is based on experimental data, additional knowledge may be needed to interpret the implications of the differences between the experimental situation and the current situation or site conditions.

If a relevant experience is not close enough to the current site conditions, the relevant knowledge may be used in the form of a holistic model that computes a corrosion rate through first principles. When used as a basis for memory construction, the corrosion rate is computed through a series of process models or database lookups of collected field data as shown in Figure 2.5.

During the course of memory construction, the environmental conditions are reevaluated according to the relevant experience used for memory construction. The relevant experience can shift the focus to different aspects of the environment according to the critical features of the selected experience and thus introducing new specifications. New indices to the selected experience are created when the experience is found to be applicable to similar situations and when the experience is restructured according to the interactions with the environment.

Figure 2.5 Computation of corrosion rate through a series of process models and parameters obtained from database lookups

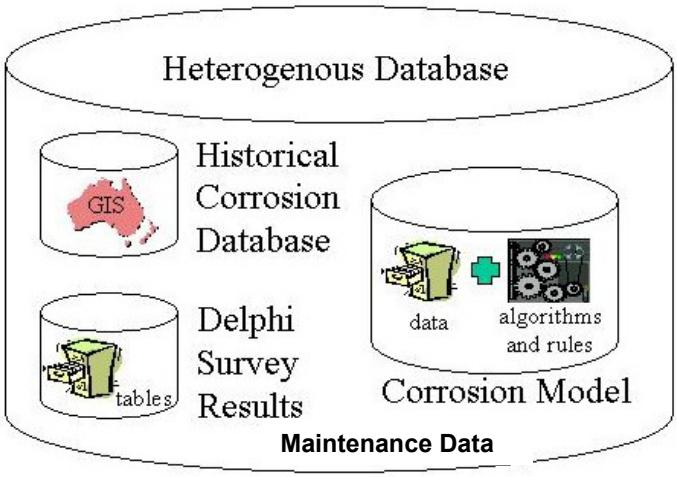


3. SITUATED CBR FOR PREDICTION OF CORROSION RATES

3.1 Components

The aim of the software tool being developed is to facilitate the accessing of a range of sources of information about corrosion and lifetime estimates for building materials. Thus the components are the various data sources (databases) and the case-based reasoning engine that will provide the linkage between them and reasoning ability to choose the appropriate instances from the various examples. The various sources of information available to the project and included in the software tool are displayed in Figure 3.1 and described in the following sections.

Figure 3.1 Heterogeneous database with four different sets of data



3.1.1 Delphi Database

The CRC for Construction Innovation has developed a database of predicted lifetimes for a range of metallic building components derived from expert opinion in the project 2002-010-B. A detailed description of this project and its outcome can be found in the final report for the project. (Cole et al., 2004)

3.1.1.1 The Delphi Technique

A Delphi survey is a structured group interaction process that is directed in 'rounds' of opinion collection and feedback. Opinion collection is achieved by conducting a series of surveys using questionnaires. The result of each previous survey will be the basis of the formulation of the questionnaire used in the next round. The Delphi technique is an established method for obtaining consensus and has been used in a variety of professional settings.

Professionals such as builders and architects were the primary respondents to the survey. They were selected on the basis of their practical experience and theoretical knowledge. Building material suppliers were also invited to participate in the survey for their intimate

knowledge of their specific products. Academics and scientist were also included because it is believed that they understand scientific principles in areas that are related to material durability, and so their expertise was relevant to the construction of a durability model. The survey was conducted via the Web to allow respondents to answer questions at a time convenient to them.

The survey included both service life (with and without maintenance) and aesthetic life, and time to first maintenance. It included marine, industrial and benign environments and covered both commercial and residential buildings.

3.1.1.2 Classification of Responses

Respondents were asked to gauge the life expectancy of a range of building materials in the different categories with answers given in year brackets: <5, 5-10, 10-15, 15-20, 20-30, 30-50, and >50 years. Responses were analysed and classified as to the level of agreement found between respondents. A class 1 response had more than 50% of answers in one year bracket, a class 2 response had more than 50% in two adjacent year brackets, a class 3 response had more than 50% of responses in three adjacent year brackets and a class 4 response showed little agreement. Examples of class 1 and class 2 responses are shown in Figure 3.2 and Figure 3.3.

Figure 3.2 An example of a class 1 response in the Delphi survey

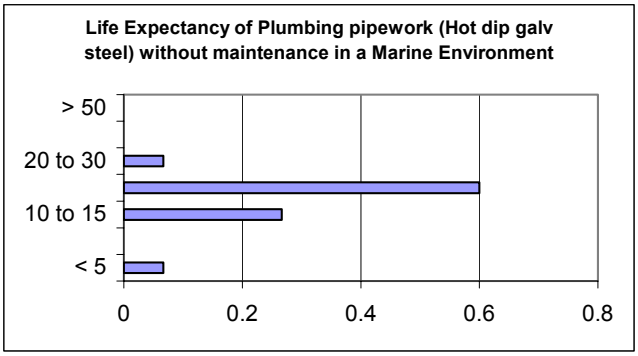
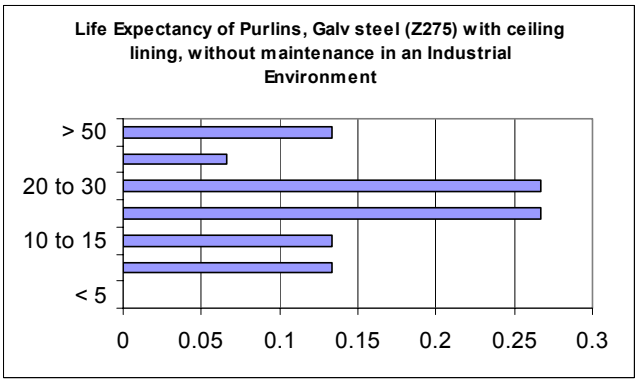


Figure 3.3 An example of a class 2 response in the Delphi survey



Class 3 and 4 responses were used as the basis for the second round of questions to try to reduce the level of uncertainty. After assessing all answers and considering the level of agreement amongst respondents, the predicted life was stored in the database in two forms: the mode and the mean as well as a standard deviation for the mean. Around 85% of all answers were fell in class 1 or 2 and were considered acceptable for inclusion in the database.

Components covered by the database are a representative subset of building materials ranging from nails and ducting through to roofing, window frames and door handles. Not all situations are covered as the components were limited to 30 and only those situations where good agreement between the experts was found were included in the database. An example of the information stored in the database is given in Appendix I.

3.1.1.3 Validation of the Delphi Database

The final database was examined in three ways to determine its accuracy and reliability. These were:

- analysis for internal consistency of the data (eg. would expect similar results from residential and commercial buildings),
- analysis for consistency with expected trends based on knowledge of materials performance (eg. stainless steel should last longer than galvanised steel) and environmental severity (eg. a roof in a benign location should last longer than one in a marine location), and
- correlation with existing databases on component performance.

In all of these comparisons, the Delphi survey data showed good agreement.

3.1.2 Queensland Department of Housing - Maintenance Database

CSIRO has, in conjunction with the Queensland Department of Housing (outside this project), analysed over 10,000 records with regards to significant maintenance. A sample of the data is presented in Appendix II. In Table 3.1 a summary of the average period to roof replacement or significant repair is given for domestic houses in southern Queensland.

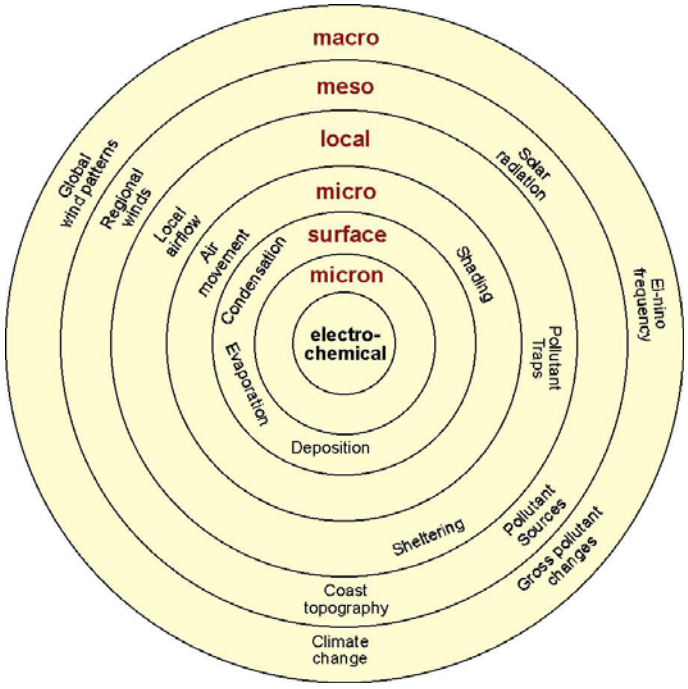
Table 3.1 Summary of the average period to roof replacement or significant repair for domestic houses in Southern Queensland

Environment	Mean (Years)	SD (Years)
Marine	16	5
Benign	41	4

3.1.3 Holistic Model

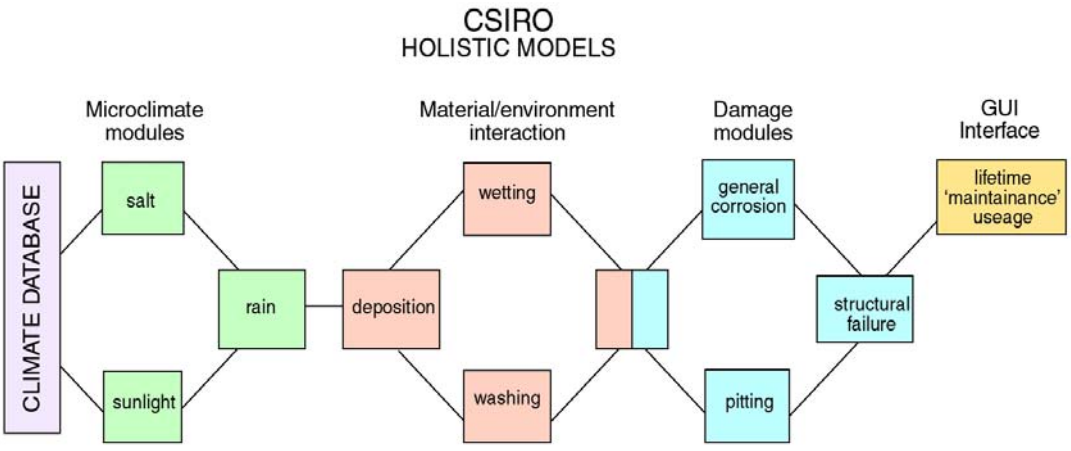
Through many years of research, CSIRO has developed a holistic model for corrosion which is based on an understanding of the basic corrosion processes ranging in scale from atomic electrochemical reactions to the macro scale of continental environmental factors (Figure 3.4).

Figure 3.4 A representation of the many factors considered in the holistic model for corrosion



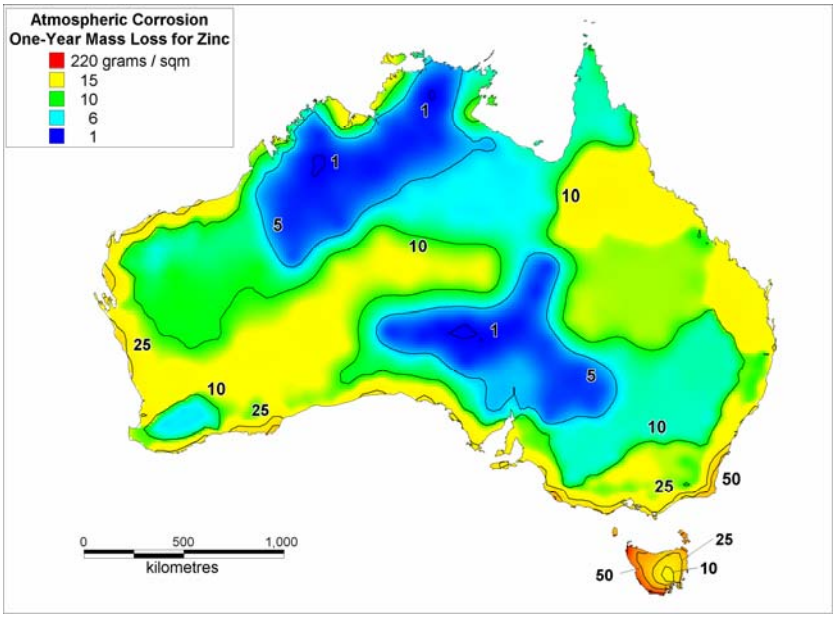
The overall model consists of three broad groups of modules: microclimate models, material/environment interactions and damage or corrosion models. These are illustrated in Figure 3.5.

Figure 3.5 Structure of the modules of the holistic model



The model starts from an understanding of climatic conditions pertinent to corrosion that include moisture, prevailing winds, salinity and pollution. This has been used to produce a Geographical Information System which models sources and distribution patterns of natural and man-made pollutants across Australia and combines this knowledge with an understanding of the physical responses of surfaces. Corrosion maps of Australia have been created from this model. This is illustrated for zinc in Figure 3.6.

Figure 3.6 Levels of corrosion of zinc due to atmospheric pollutants



The holistic model forms another source of information for predicting the lifetime of metallic building components. Once the geographical position is specified, salt deposition levels can be determined. Modifications can be made according to the type and positioning of the element (see case definition below) and how this affects the climatic factors.

It may also be necessary in some applications to carry out further modifications to the algorithms of the holistic model to allow more accurate calculations to be made. This was done for the gutter application presented in this report (and discussed in Section 5.2).

3.2 Definition of Cases

Intrinsic to the use of case-base reasoning is the definition of the attributes for cases for a particular application. Cases need to be defined such that the CBR engine can search the casebase and other databases for examples relevant to the current case. In the application for lifetime prediction, parameters relevant to component degradation need to be considered and defined.

For metallic building components the important parameters for determining the corrosion rate include the component type, material type and the environmental conditions. A detailed review of corrosion degradation models for metallic components used in building materials was carried out and is summarised in CRC Report 2002-059-B No. 5 “Corrosion Degradation Models for Metallic Building Components”.

Materials in common use include:

- steel and steel alloys (bare and painted)
- zinc and zinc alloys (bare and painted)
- aluminium alloys.

The parameters identified to control corrosion degradation rates are summarised as:

- Time of wetness
- Chloride concentration
- Sulfur dioxide concentration (or deposition of other sulfur impurities)
- Ozone concentration
- Temperature
- pH of precipitation
- Volume of precipitation
- Deposition of dust
- Nitrogen oxide (NO_x) concentration

These parameters are all strongly dependent on geographic location, with climate and local industry level of paramount importance. Once the macroclimate has been identified then the rate of corrosion will also depend on placement within the building eg internal or external, if external then whether sheltered or exposed etc. A final parameter of importance is whether the building element is subject to regular maintenance. Maintenance includes cleaning and repainting but does not extend to replacement of the building component.

3.2.1 Characterisation of Environment

For the purpose of corrosion the environment needs to be characterized in terms of the pollutant, RH and type of rainfall. This is summarised in Table 3.2.

Table 3.2 Environment classifications

Pollutant	**RH**	**Rainfall**
Severe Marine	Very Humid	Frequent and Heavy
Marine	Humid	Frequent and Light
Severe Industrial	Standard	Standard and Heavy
Moderate Industrial	Standard	Standard and Light
Industrial	Dry	Infrequent and Heavy
Benign	Very Dry	Infrequent and Light

In addition, for the severe marine, marine, severe industrial and industrial classifications the neighbourhood must also be considered with the following classifications based on how the surrounding land use affects pollutant transport:

- Grassland,
- Urban,
- Forest,
- High rise

3.2.2 Detailed Building Characterisation

3.2.2.1 Location in Building

Building structures have been considered with regard to the situations that will affect the amount of aerosol deposition of pollutants. Thus building locations have been divided into twelve types and these are listed in Table 3.3.

Table 3.3 Description of building elements for case definition

Case	Description
Open Rooftop	The top of any surface that bridges between the tops of two or more walls and has an average slope of 45 degrees or less. This includes flat, hip, gable, monoslope, multispan, sawtooth, arched mansard and conical roofs. It includes projections and indentations of 0.3 metres or less. The roof is to have a minimum dimension of at least two metres.
Open Wall	Any flat non-sheltered surface with a slope of less than 45 degrees off vertical including any projections or indentations that depart less than one metre from planarity. The wall is to have minimum dimension of at least one metre. Also includes bridge piers.
Sheltered Wall	Any area that is covered with a covering that stops all direct sunlight when the sun is less than 45 degrees from the zenith
Edges and External corners of walls or roofs	Comprises the area within one metre of any external corner. This excludes re-entrant corners, corners on isolated steelwork, and corners on some roofs (such as saw-tooth roofs). The angle of the external corner is to be between 0 and 135 degrees. It includes corners of bulk objects projecting from roofs.
Dirt Accumulation Zone	Any area in which water, dirt, leaves or dust can accumulate. This surface usually has an angle of less than 3 degrees to the horizontal but as corrosion develops it can grow to encompass much steeper angles
Roof cavity	Any object lining or found within the cavity between the ceiling and roof of a building.
Wall cavity	Any object lining or found within the cavity between the inner and outer walls of a building. Also includes cavities in multistorey buildings between the false ceiling and the floor above.
Moisture Accumulation Points in Wall Cavities	e.g bottom Plates
Underfloor cavity	Any object lining or found within the space under the ground floor of a building. Excludes any such space that is artificially heated or ventilated.
Semi-enclosed space	Seem most frequently as a lower floor in a multistorey car park. Defined as any object in a space with at least one large opening to the atmosphere. Excludes any such space that is artificially heated or ventilated.
Enclosed room	Includes rooms in domestic residences, commercial establishments, factories and warehouses, and elsewhere. Estimating the corrosion in an enclosed room requires further information on heating, artificial ventilation, and local sources of aerosols, gases and moisture.

3.2.2.2 Cleaning

Corrosion is also affected by how much of any pollutant deposition can be removed by the natural cleaning of rain, condensation and wind. Classifications with regard to cleaning levels are listed in Table 3.4.

Table 3.4 Definitions of cleaning

Case	Description
Open Rooftop	Any area exposed to sun and rain with a slope between 3 degrees and 45 degrees (but see (6) below
Open Wall	Any area that is not sheltered with a slope of less than 45 degrees off vertical.
Sheltered	Any area that is covered with a covering that stops all direct sunlight when the sun is less than 45 degrees from the zenith.
Crevice	Any gap small enough for capillary attraction to drag water upwards
Drop-off Zone	Any area from which water will drop. This typically occurs under the edges of overhangs
Dirt Accumulation zone	Any area in which water, dirt, leaves or dust can accumulate. This surface usually has an angle of less than 3 degrees to the horizontal but as corrosion develops it can grow to encompass much steeper angles.

3.2.2.3 Maintenance

If a metallic building element is subject to a regular maintenance schedule that will pick up and deal appropriately with the first signs of corrosion, then it is likely to last longer than one that is not maintained in the same situation. Thus maintenance (or lack of) is considered an important parameter for definition of a case. It is particularly an issue for building components, such as gutters, where dirt and debris can collect over time and affect drainage and the rate of drying after rainfall or condensation.

3.3 Defining Case Similarity

When the liftetime prediction tool is presented with a new case, it will search through the casebase library to find similar cases that have already been constructed. Whilst it is possible that a stored case may exist that matches all the case parameters exactly, it is more likely that some variation will occur. Thus it is necessary to have some method of defining how similar the new case is to each of those stored in the casebase and extracting the cases considered most 'similar'.

Similarity between cases must be based on similarity in the attributes that affect the corrosion rate of the building materials under consideration ie.

- Geographic location
- Location in Building
- Maintenance, and
- Cleaning

Overall, a similarity number (S) will be defined, where:

$$S = M_s \times C_s \times L_s \times G_s \qquad \text{… Eqn 3.1}$$

Where:

M_s is a measure of similarity in Maintenance, the Maintenance similarity index,

C_s is a measure of similarity in Cleaning, the Cleaning similarity index

L_s is a measure of similarity in Location in Building, the Location similarity index and

G_s is a measure of similarity in geographic location, the geographic similarity index.

If two parameters match exactly, the similarity index will equal 1. If two parameters are different but have similar effects on the likely corrosion rate, then the similarity index will be close to 1 (0.8-0.9). The lower the similarity index, then the greater the difference between the two situations in terms of likely corrosion rates. Since the individual similarity indices are multiplied together to provide the overall similarity index S, variations in individual indices result in a cumulative lowering of S. The cut-off point for S at which a case is not retrieved from the casebase can be defined to broaden or narrow the cases chosen.

Values for the similarity indices have been defined and are discussed in the following sections. At this stage of the project development, not all cases have been considered, so only a subset (relevant to the gutter application) are presented.

3.3.1 Maintenance

At this stage there are only two values for Maintenance: maintained or not maintained. For comparison between cases, the values in Table 3.5 are assigned for M_s, the maintenance similarity parameter.

Table 3.5 Values defined for Maintenance Similarity Index Ms

New Case / Old case	Maintained (M1)	Not Maintained (M0)
Maintained (M1)	1	0.7
Not Maintained (M0)	0.7	1

3.3.2 Cleaning

The most important aspect for cleaning is considered to be whether or not the building component is in a dirt accumulation zone. If two cases do not match in this aspect then C_s = 0. If they do match then Cs is assigned values according to Table 3.6 and Table 3.7, depending on whether maintenance is available or not.

Table 3.6 Values of Cs for non-dirt accumulation zone

New Case / Old case	Maintained (M1)	Not Maintained (M0)
Maintained (M1)	1	0.9
Not Maintained (M0)	0.9	1

Table 3.7 Values for Cs in dirt accumulation zone

New Case / Old case	Maintained (M1)	Not Maintained (M0)
Maintained (M1)	1	0.7
Not Maintained (M0)	0.7	1

3.3.3 Location in Building (L_s)

For the building components defined in Section 3.2.2.1 the similarity index Ls is defined in Table 3.8.

Table 3.8 Values of Ls for Building Components

	Building Location - New case											
Old case	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
1	1	0.8	0.7	0.7	0.6	0.5	0.5	0.6	0.6	0.7	0.7	0.5
2	0.8	1	0.8	0.7	0.6	0.5	0.5	0.6	0.6	0.7	0.7	0.5
3	0.7	0.8	1	0.8	0.6	0.5	0.5	0.6	0.6	0.7	0.8	0.5
4	0.7	0.7	0.8	1	0.7	0.5	0.5	0.6	0.6	0.7	0.8	0.5
5	0.7	0.6	0.6	0.7	1	0.5	0.5	0.6	0.6	0.8	0.7	0.5
6	0.5	0.5	0.5	0.5	0.5	1	0.9	0.8	0.7	0.5	0.5	0.8
7	0.5	0.5	0.5	0.5	0.5	0.9	1	0.9	0.8	0.6	0.6	0.7
8	0.6	0.6	0.6	0.6	0.6	0.8	0.9	1	0.9	0.7	0.7	0.6
9	0.6	0.6	0.6	0.6	0.6	0.7	0.8	0.9	1	0.8	0.8	0.5
10	0.7	0.7	0.7	0.7	0.8	0.5	0.6	0.7	0.8	1	0.7	0.5
11	0.7	0.7	0.8	0.8	0.7	0.5	0.6	0.7	0.8	0.7	1	0.6
12	0.5	0.5	0.5	0.5	0.5	0.8	0.7	0.6	0.5	0.5	0.6	1

3.3.4 Geographic Location (G_s)

The most important aspect for geographic location is considered to be whether or not the specified case is in a marine environment or not (benign, salinity < 15 mg/m^2.day). If two cases do not match in this aspect then $G_s = 0$.

If two cases being compared are both non-marine then:

$G_s = 1$ if they are within 20 km of each other, and

$G_s = 0.9$ if they are within 50 km of each other.

For two marine cases or non-marine cases > 50 km apart then G_s is assigned values according to:

$$G_s = W_s * M_s \qquad \text{…Eqn 3.2}$$

where Ws is the time of wetness similarity factor (defined inTable 3.9) and Ms is the marine salinity factor (defined in Table 3.10).

Table 3.9 Values for Ws (Time of Wetness similarity index)

	New case-TOW (%)				
Old case-TOW(%)	**0 to 19**	**20-39**	**40-59**	**60-79**	**80-100**
0 to 19	1	0.8	0.7	0.6	0.5
20 to 39	0.8	1	0.8	0.7	0.6
40 to 59	0.7	0.8	1	0.8	0.7
60-79	0.6	0.7	0.8	1	0.8
80-100	0.5	0.6	0.7	0.8	1

Table 3.10 Values for Ms (marine salinity similarity index)

	New case-Salinity mg/m^2.day					
Old case-Salinity mg/m^2.day	**<5**	**5 – 15**	**16-40**	**41-100**	**101-300**	**>300**
<5	1	0.8	0.7	0.6	0.5	0.4
5 -15	0.8	1	0.8	0.7	0.6	0.5
16 40	0.7	0.8	1	0.8	0.7	0.6
41 -100	0.6	0.7	0.8	1	0.8	0.7
101 - 300	0.5	0.6	0.7	0.8	1	0.8
> 300	0.4	0.5	0.6	0.7	0.8	1

3.3.4.1 Computation of Distance between Two Points on the Earth's Surface

Due to the near spherical shape of the Earth (an oblique spheroid), spherical geometry and trigonometric mathematical functions are required to calculate an accurate distance between two points on the surface of the earth. The equations are given in Appendix III.

4. DESIGN AND IMPLEMENTATION OF SITUATED CBR FOR CORROSION PREDICTION

4.1 Specification and Design

4.1.1 Definition of Problem

The aim of the project is to improve the accuracy of component life prediction through the use of a situated CBR system that operates upon its knowledge and experience. The holistic model will be used to provide the required knowledge for computing the life building components through first principles. A number of databases of component life derived from expert opinion, research and maintenance data will provide a repository of experiences on component life predictions of certain components under specific conditions. A case base of previously constructed memory will provide a repository of experiences that predict component life based on combining the results from the holistic model and databases.

The project has focused on the creation of the software architecture for component life prediction based on situated CBR. The architecture provides a structure for utilising existing knowledge and experiences to reason about the current situation (interpretation) and construct a solution to component life prediction (construction).

The software has been designed with consideration for the two applications specified by the project's industry partners:

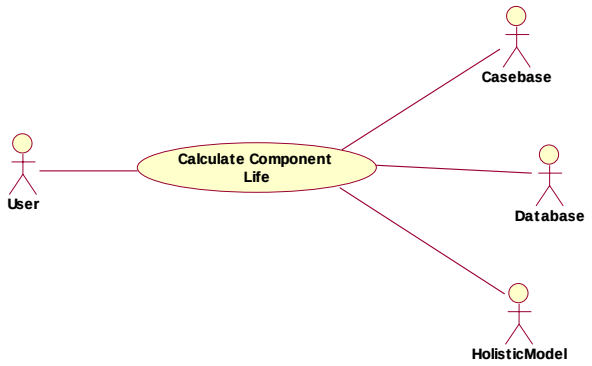
- Gutters in Queensland schools for the Queensland Department of Public Works, and
- Bridges in Queensland roads for the Queensland Department of Main Roads.

The design of this situated case-based reasoning framework is outlined in the next sections, followed by details of its implementation.

4.1.2 Usage Scenario

Figure 4.1 illustrates the overall usage scenario for the proposed system by an external user. The way the system is used is outlined.

Figure 4.1 Use case outlining the overall usage of the system



The user of the system supplies the following information (items in parenthesis indicate their possible values):

- Location of Site (coordinates pair on longitude and latitude in decimal degrees)
- Type of Component (roof / gutter)
- Material of Component (galvanized steel / zincalume / Colorbond)
- Maintenance State (maintained / not maintained)
- Cleaning Condition (dirt can collect / dirt cannot collect)
- Cleaning State (cleaned / not cleaned)
- Location of Component within Building (list of locations)
- Condition of Geographic Location (marine application / non marine application)

The list of locations consists of:

- Open Rooftop
- Open Wall
- Sheltered Wall
- Edges and External Corners of Walls or Roofs
- Dirt Accumulation Zone
- Roof Cavity
- Wall Cavity
- Moisture Accumulation Points in Wall Cavities
- Underfloor Cavity
- Underfloor Positions in Contact with Earth
- Semi-Enclosed Space
- Enclosed Room

Based on the input supplied, the situated CBR system computes a predicted component life value from its Casebase, Database and HolisticModel. The Casebase is a repository of previous prediction episodes and the HolisticModel is a set of procedures that calculate the component life of building elements. Database is a placeholder that represents different databases that contain component life information from different sources.

4.1.3 System Scenario

The usage scenario in Figure 4.1 is mapped onto a *system scenario* illustrated in Figure 4.2. This (system) scenario outlines the way in which a situated CBR system handles the problem presented in the use case of Figure 4.1. Table 4.1 provides details to this system scenario.

Figure 4.2 Details of Calculate Component Life use case

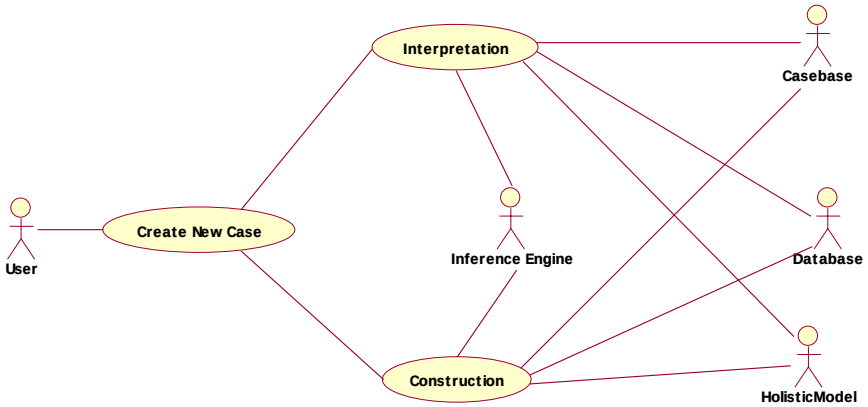


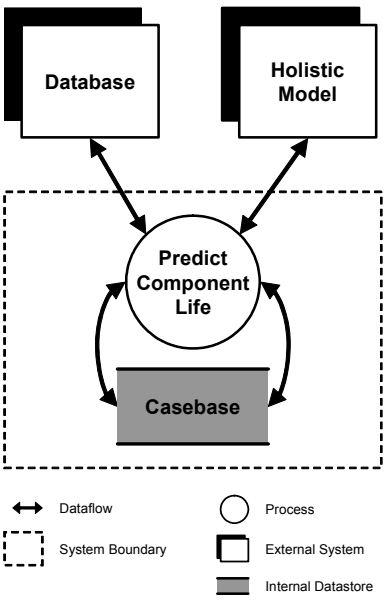
Table 4.1 Details of system scenario

Descriptions	This scenario illustrates different processes within the situated CBR system used to predict component life.
Actors	User, Casebase, Databases (delphi and field (maintenance) databases) and Holistic Model.
Preconditions	System initialized with the required data for computing similarity indices.
Scenario Text	Activity: User enters values for the following parameters to start the creation of a new case: Location of Site Type of Component Material of Component Maintenance State Cleaning Condition Cleaning State Location of Component within Building Condition of Geographic Location Interpretation based on initial parameter values: The casebase, databases and holistic model are accessed to generate alternatives (subcases) for finalizing the inputs. An inference engine is employed to process these alternatives to produce this finalization. Construction based on finalized parameter values: The finalized inputs are used again to generate another set of alternatives from the casebase, databases and holistic model. An inference engine is used to combine these alternatives to produce a complete solution for component life prediction. The current problem solving episodic is stored as a new case.
Alternative Courses	None
Extends	None
User Interfaces	None
Constraints	None
Questions	None
Notes	Both inference engines for interpretation and construction need not be implemented but provisions must be made for their incorporation later.
Author	Pak-San Liew
Source Document	System Specification Report (ver. 1.0)

4.1.4 Context Diagram

To define the scope of the situated CBR system, a context diagram and usage scenario are employed. Figure 4.3 illustrates the boundaries of the situated CBR system in the form of a context diagram. . The Database box denotes different physical databases of component life information obtained by different means. For the current system, two databases: one based on a Delphi study of experts' opinions (*Delphi database*) and one based on field (maintenance) data (labeled as *field (maintenance) database* in this project), are considered.

Figure 4.3 Context diagram for the situated CBR system



The development of the system entails the creation of the following:

- interfaces to different database;
- interface to a casebase to store previous problem solving episode;
- a software framework to contain the above and
- different entry points within the framework to allow the incorporation of different inference engines for interpretation and construction as defined in situated CBR.

4.1.5 Software Environment

The situated CBR system is expected to operate within the Windows XP environment utilizing Java version 1.4.

4.1.6 System Architecture

The basic goal of this project is to develop a software framework for component life prediction based on the use of knowledge and past experiences according to the model of situated CBR (Liew and Maher 2004). The architecture of a situated CBR system is the main focus. This architecture provides a structure that defines the infrastructure that permits situated case-based reasoning. *What software components* are involved and their *interconnections* are emphasized here. Development of the internal workings of different components within the architecture that are dictated by the domain of corrosion engineering is not within the scope of this project.

To achieve the architectural goal, a prototype system is created to define a framework for situated CBR within a software system. The architecture of this system:

- provides the infrastructure for reasoning about the current situation according to its knowledge and experiences during interpretation; and

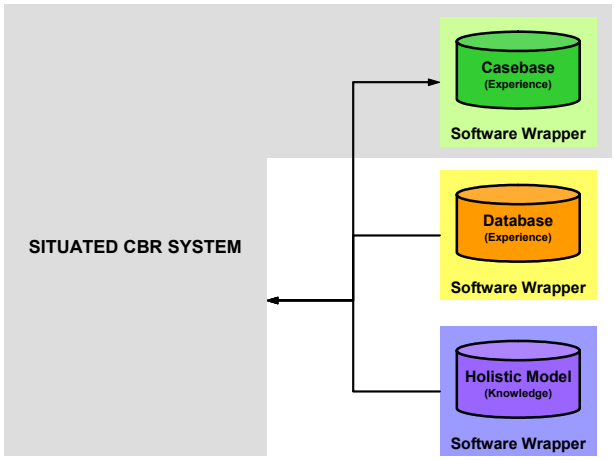
- provides the infrastructure for reasoning about the ways solutions from casebase, the holistic model and various database can be combined to produce a complete solution

4.1.6.1 Software Architecture

Figure 4.4 illustrates the overall software architecture of the prototype system. All codes reside within a single machine and no distributed computations are considered in the design of the system.

Software wrappers are used to insulate data storage technologies from the situated CBR system. A wrapper encapsulates the details of an underlying persistence technology through an interface. This interface provides a set of common access methods to the required data across different persistence technologies. Components of the situated CBR system that require data storage and retrieval functionalities are only required to conform to the method signatures of the relevant interface without being concerned about the technology used. When the technology is changed in subsequent development of the system, changes to the situated CBR system are isolated to the backend of the wrapper that interacts directly with the new technology. Codes within the situated CBR that utilizes the wrapper are not affected.

Figure 4.4 Software architecture of the situated CBR system



4.1.6.2 Solution Approaches for Interpretation and Memory Construction

Error! Reference source not found. illustrates the key components that will implement the interpretation and construction of memory for the situated CBR system. During interpretation (Figure 4.6), the interpreter uses the input values from the user to retrieval alternatives from the casebase, databases of component life and holistic model to finalize the input values entered by the user. For the database of experience, two databases are considered currently: a Delphi database of experts' opinions and a field (maintenance) database of measurements. During construction (Figure 4.7), alternatives are retrieved from the casebase, databases of component life and holistic model in a similar way according to the finalized input to provide the basis for constructing the solution to the current prediction problem.

Figure 4.5 Key software components for interpretation and memory construction

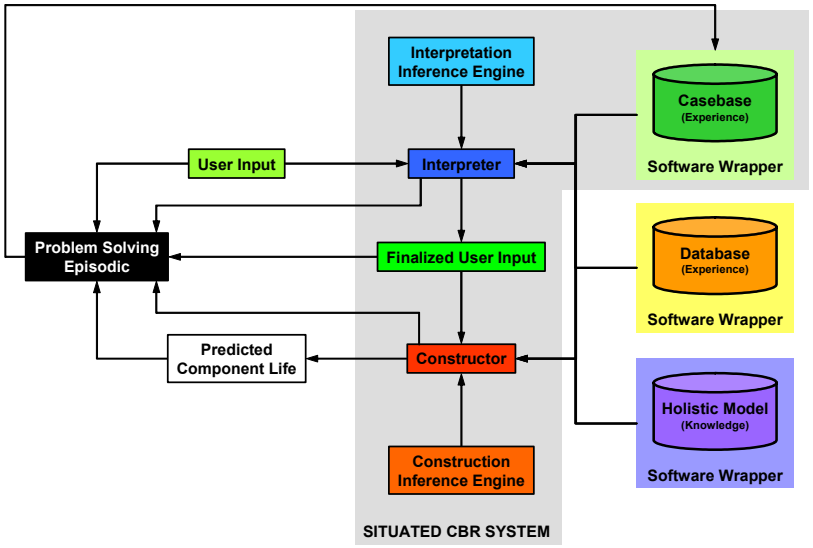


Figure 4.6 Interpretation cycle

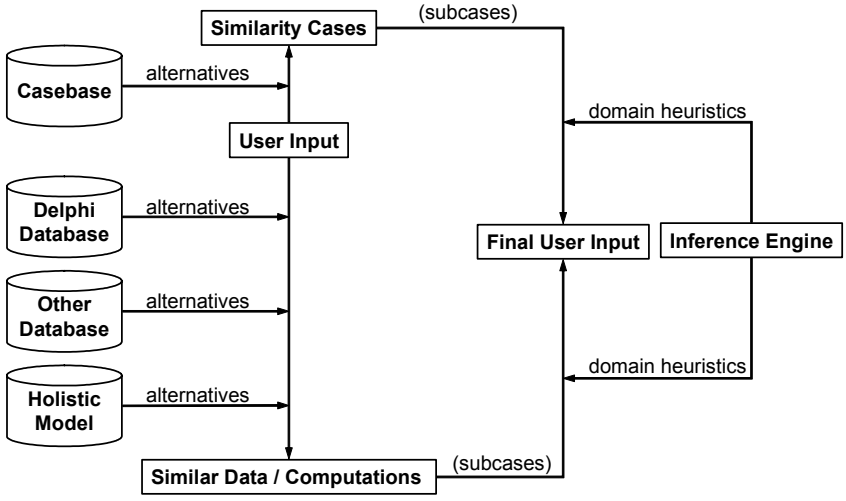
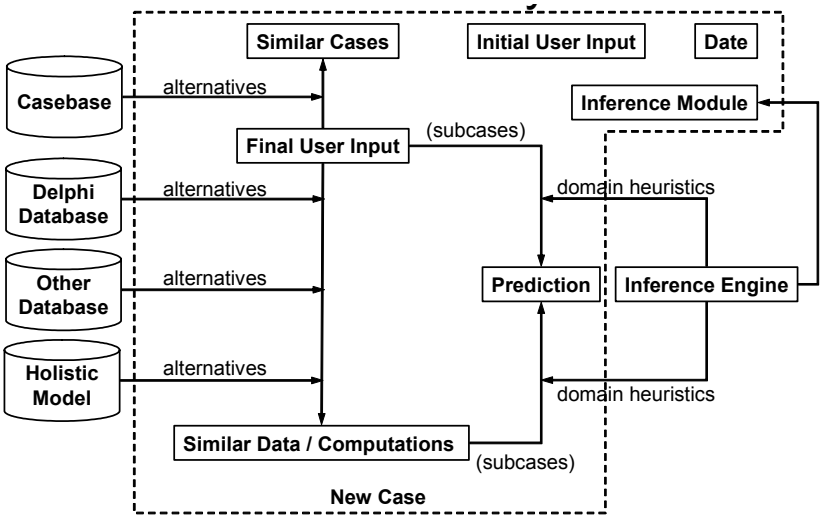


Figure 4.7 Construction cycle



4.1.6.2.1 Contents of a Case

The end result of using the system is a new case. This case is made up of the following:

- initial user input;
- finalized user input;
- alternatives (subcases) used in construction consisting of: similar cases from the casebase, similar data from the Delphi and field databases, as well as similar computations from the holistic model;
- prediction component life;
- time stamp; and
- an inference module indicating how the prediction value is computed.

Information from interpretation is not stored.

4.1.6.2.2 Generation of Alternatives with Cases from the Casebase

In terms of information from the casebase, associated cases of previous prediction episodes are retrieved so that previous problem solving experience can be utilized. A retrieved case is defined as similar to the current situation when its similarity index is computed to be **>0.5**. This value is set arbitrarily for the current development and can be fine-tuned later.

For the Queensland schools gutter application the similarity parameters have been defined. In situations where the **cleaning condition** of the user input and the retrieved case are different (one with **gunk can collect** and the other with **gunk cannot collect**), the cleaning factor (C_S) is set to zero so that the retrieved case will not be used.

In situations where the conditions of **geographic-location condition** of the user input and the retrieved case are different (one with **marine application** and the other with **non marine application**, the geographic-location factor, G_S, is set to zero so that the retrieved case will not be used.

For the geographic-location factor, G_S, with **non marine application** condition, if the location that was entered by the user and the retrieved case are within 20 km of each other, G_S is set to the value of **1.0**. If the two locations are between 20 to 50km, G_S is set to **0.9**. Any distance greater than 50km sets G_S to **0.0**.

4.1.6.2.3 Generation of Alternatives with Data from Databases

Experiences in terms of datum from the databases (Delphi and field) that are retrieved are based on the use of retrieval key values as entered by the user and alternative values that certain keys can take. For example, if the user entered a value of **Galvanized Steel** in the material field of the user input, this value for material is used as part of a series of keys to retrieve component life data from the Delphi and field (maintenance) databases. To retrieve alternative component life, the system changes the material field to **Zincalume** (while maintaining the same values for other fields) to retrieve more data from these databases.

Currently, only the material field of the user input is allowed to change and the system cycles through all material in the databases to generate alternatives. Expansions in terms of additional materials can be added to the wrappers of the individual data source.

4.1.6.2.4 Generation of Alternatives with Data from the Holistic Model

Alternatives from the holistic model are generated in a way similar to that for databases: all available materials are used as alternative input to generate different component life data.

4.1.6.2.5 Interpretation from Alternatives

The interpretation process is instantiated as a finalization of input parameter values from the alternatives generated from above. These alternatives are modelled within the system as subcases that provide optional values for variable input parameters.

In the current development, the use of different materials to retrieve data provides the user with alternatives in terms of using different materials for the same situation. These alternatives provide the basis for reframing the problem specification by modifying the initial input values entered by the user to finalize the input values.

An inference engine is employed for the finalization process. The intelligence for this process is currently implemented by displaying the alternatives and prompting the user for a solution.

4.1.6.2.6 Memory Construction from Alternatives

Based on the finalized input data, alternatives for memory construction are generated from the casebase, databases and holistic model in the same way as in interpretation. Another inference engine is employed to combine these results and construct a complete solution for predicting component life.

As in the case of interpretation, the inference engine is not implemented in the current system but an entry point within the system's architecture has been provided. The intelligence for construction is currently implemented by displaying the alternatives and prompting the user for a solution.

4.1.6.2.7 Hard-Coding of Interpretation and Construction

To emulate the operations of the inference engines employed during interpretation and construction, dummy functions are used as placeholders for function calls to these engines. During interpretation, the alternatives from the casebase, databases and holistic models are displayed as subcases and the initial input values are duplicated to create the finalized values. During memory construction, the alternatives from the casebase, databases and holistic models (based on the hard-coded input values from interpretation) are also displayed as subcases and the final prediction value is set to **100.0**. The module parameter that indicates how the reasoning was done is set to a value PLACEHOLDER.

4.1.6.3 Solution Approach for Calculating Similarity Index

Comparison of the current case with information (experience) stored in the various databases is integral to the case-based reasoning program. Therefore a method must be found for quantifying the similarity index of different cases.

Information for the computation of similarity matrix (See Section 3.3) is encoded within an ASCII file (Figure 4.8). During initialisation, this file is read into the system to allow similarity indices to be calculated.

The use of an ASCII file to represent the tabulated data in the similarity tables is to allow the easy modification of their values in the future.

4.1.6.4 Rationale for Design Decisions

Design decisions that have consequence on the behaviours of the system are outlined in the following table (Table 4.2).

Table 4.2 Design decisions and their rationale

Design Decision	Rationale
If there is no data available within a database based on: (a) the set of original input parameter values or (b) a modified version of these values for generating alternatives during interpretation and construction, the alternative generated will have a computed value of zero to indicate this situation.	The user is informed about the fact the database do not have data for the original or modified set of input values.

4.1.6.5 Extension and Modifications Points

The following components of the situated CBR system are modification and entry points for extending the functionalities of the system:

- wrapper for casebase, database and holistic model;
- interpreter access to an inference engine;
- constructor access to an inference engine;
- system access to different tabulated data for similarity computation; and
- system access to different information for similarity computation.

The wrapper for casebase and database permits system modifications and extensions in terms of:

- using alternative persistence technologies such as a flat file system, rational or object databases for data storage and retrieval; and
- incorporating new mechanisms for generating alternatives during interpretation and construction.

Figure 4.8 ASCII file coding of the data required for computing similarity index

```
PARAMETER_START
    Name = maintenance state
    Condition = NULL
        Variable = maintained
        Variable = not maintained
        Table
        1.0     0.7
        0.7     1.0
PARAMETER_END


PARAMETER_START
    Name = cleaning
      Condition = gunk cannot collect
        Variable = cleaned
        Variable = not cleaned
        Table
        1.0     0.9
        0.9     1.0
PARAMETER_END


PARAMETER_START
    Name = cleaning
    Condition = gunk can collect
        Variable = cleaned
        Variable = not cleaned
        Table
        1.0     0.7
        0.7     1.0
PARAMETER_END


PARAMETER_START
    Name = location in building
    Condition = NULL
        Variable = open rooftop
        Variable = open wall
        Variable = sheltered wall
        Variable = edges and external corners of walls or roofs
        Variable = dirt accumulation zone
        Variable = roof cavity
        Variable = wall cavity
        Variable = moisture accumulation points in wall cavities
        Variable = underfloor cavity
        Variable = underfloor positions in contact with earth
        Variable = semi enclosed space
        Variable = enclosed room
        Table
        1.0   0.8   0.7   0.7   0.6   0.5   0.5   0.6   0.6   0.7   0.7   0.5
        0.8   1.0   0.8   0.7   0.6   0.5   0.5   0.6   0.6   0.7   0.7   0.5
        0.7   0.8   1.0   0.8   0.6   0.5   0.5   0.6   0.6   0.7   0.8   0.5
        0.7   0.7   0.8   1.0   0.7   0.5   0.5   0.6   0.6   0.7   0.8   0.5
        0.7   0.6   0.6   0.7   1.0   0.5   0.5   0.6   0.6   0.8   0.7   0.5
```

```
        0.5     0.5     0.5     0.5     0.5     1.0     0.9     0.8     0.7     0.5     0.5     0.8
        0.5     0.5     0.5     0.5     0.5     0.9     1.0     0.9     0.8     0.6     0.6     0.7
        0.6     0.6     0.6     0.6     0.6     0.8     0.9     1.0     0.9     0.7     0.7     0.6
        0.6     0.6     0.6     0.6     0.6     0.7     0.8     0.9     1.0     0.8     0.8     0.5
        0.7     0.7     0.7     0.7     0.8     0.5     0.6     0.7     0.8     1.0     0.7     0.5
        0.7     0.7     0.8     0.8     0.7     0.5     0.6     0.7     0.8     0.7     1.0     0.6
        0.5     0.5     0.5     0.5     0.5     0.8     0.7     0.6     0.5     0.5     0.6     1.0
PARAMETER_END


PARAMETER_START
    Name = time of wetness
    Condition = marine application
        Variable = 0 to 19
        Variable = 20 to 39
        Variable = 40 to 59
        Variable = 60 to 79
        Variable = 80 to 100
        Table
        1.0     0.8     0.7     0.6     0.5
        0.8     1.0     0.8     0.7     0.6
        0.7     0.8     1.0     0.8     0.7
        0.6     0.7     0.8     1.0     0.8
        0.5     0.6     0.7     0.8     1.0
PARAMETER_END


PARAMETER_START
    Name = salinity factor
    Condition = marine application
        Variable = 0 to 4
        Variable = 5 to 15
        Variable = 16 to 40
        Variable = 41 to 100
        Variable = 101 to 300
        Variable = 300 to infinity
        Table
        1.0     0.8     0.7     0.6     0.5     0.4
        0.8     1.0     0.8     0.7     0.6     0.5
        0.7     0.8     1.0     0.8     0.7     0.6
        0.6     0.7     0.8     1.0     0.8     0.7
        0.5     0.6     0.7     0.8     1.0     0.8
        0.4     0.5     0.6     0.7     0.8     1.0
PARAMETER_END
```

(Figure 4.8 cont.)

Expansions in terms of new mechanisms for generating alternatives include the addition of new parameter values for variable retrieval keys (such as more material types) and the incorporation of new variable retrieval keys (such as allowing component type to change on top of material).

The wrapper for the holistic model permits system modifications and extensions in terms of:

- using updated versions of the holistic model; and

- incorporating new mechanisms for generating alternatives during interpretation and construction (as in the case of casebase and databases).

Inference engines based on Artificial Intelligence (AI) technologies that model the required domain heuristics during interpretation and construction are accessed through the interpreter and constructor respectively. Currently, dummy function calls are used to emulate these accesses.

The information required for computing similarity is encapsulated within a component that reads off the required data from an ASCII file. For the current ASCII file, only modifications of different tabulated values within the file are permitted. For all other structural changes, new mechanisms for getting the required data need to be coded. The interface provided by the current design isolates these changes to the underlying codes that deal directly with the data. Other components of the situated CBR system that needs the information for similarity computation are not altered with these changes.

4.1.6.5 Secondary Issues

The following are taken as secondary issues during the development of the system:

- user interfaces
- extensive error handling; and
- performance and efficiency.

The structure of the software framework was seen as the main focus of the project, and simple user interfaces were developed later to facilitate the development of the applications. No provisions are also made for handling errors, performance and efficiency issues for the same reason.

4.1.7 Software Modules

The situated CBR system is decomposed into the following:

- user input module;
- display module;
- interpretation module;
- construction module;
- inference module
- similarity computation module;
- data source module and
- wrappers for data source.

Figure 4.9 presents the overall picture of how these modules relate to each other as part of the whole system. The responsibility of each module is outlined in Table 4.3.

Figure 4.9 Relationships between different modules of the situated CBR system

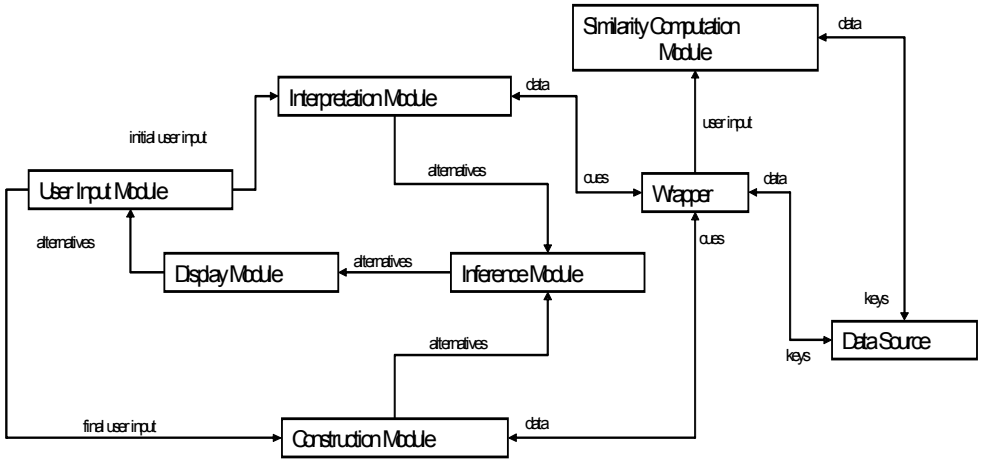


Table 4.3 Software modules in the situated CBR system

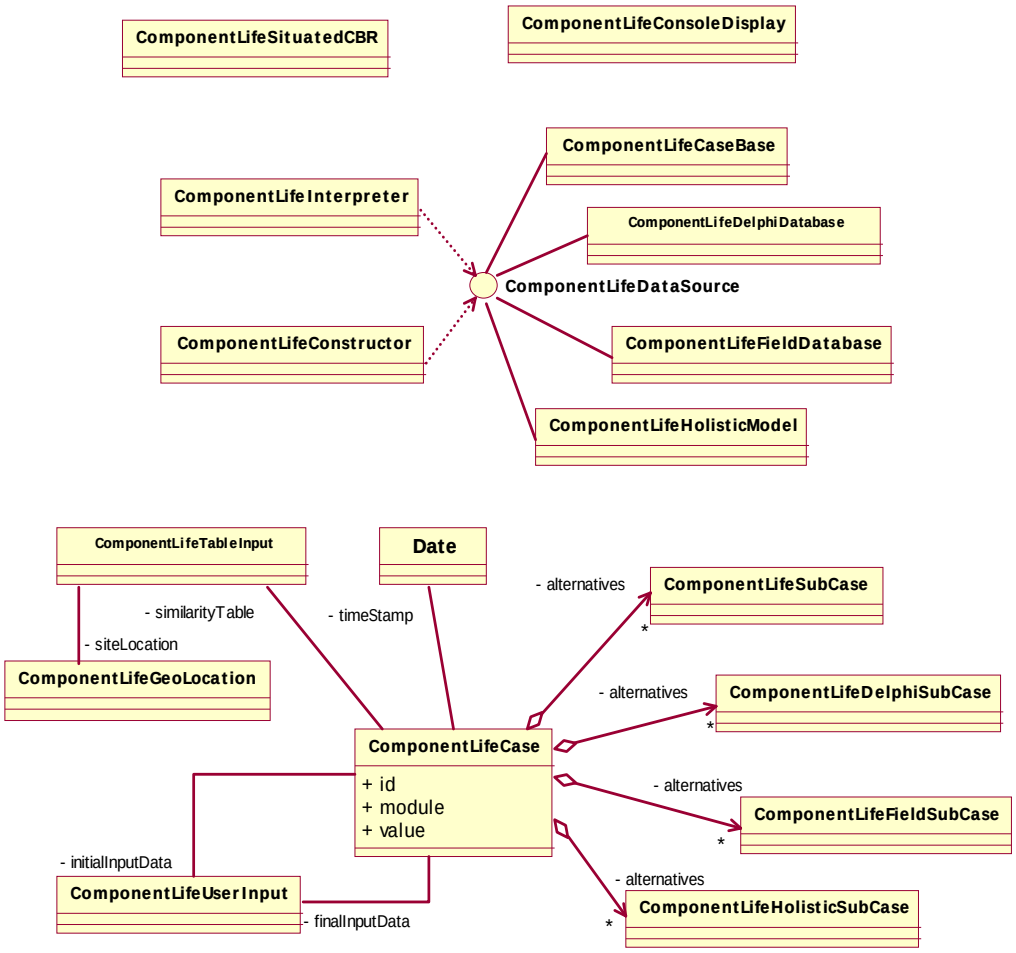
Module	Responsibility
User Input	To represent the set of user input parameters.
Display	To display the output of different processes.
Interpretation	To provide the required interpretative function for situated CBR.
Construction	To provide the required constructive function for situated CBR.
Inference	To represent domain heuristics for finalizing user input and constructing a solution from a series of alternatives. To provide an entry point for incorporating different AI engines for inferencing.
Similarity Computation	To calculate the similarity between the input parameters and previous problem solving episodes. To provide an interface to different ways to calculate similarity indices so that changes are isolated when different methods are used.
Data Source Module	To provide the required data for interpretation and construction of solutions.
Wrappers for Data Source	To provide an interface to various data sources so that changes are isolated when persistence technologies changes. To an entry point for incorporating different mechanisms that allow alternatives to be retrieved from the data sources based on variable keys.

4.2 Implementation

4.2.1 Class Diagram

Figure 4.10 outlines the key classes of the situated CBR system. Details of the classes are described in Appendix IV.

Figure 4.10 Key classes in the implementation of the situated CBR system.



4.2.2 Interactions

Interactions between different classes for a typical memory construction through interaction and construction are illustrated in Figure 4.11.

4.2.3 Extension and Modification Points

The case-based reasoning engine has been coded to allow for subsequent extension and modification. Table 4.4 details the various modification points with the relevant classes for future system changes.

Figure 4.11 Interactions between classes of the situated CBR system for a typical memory construction process

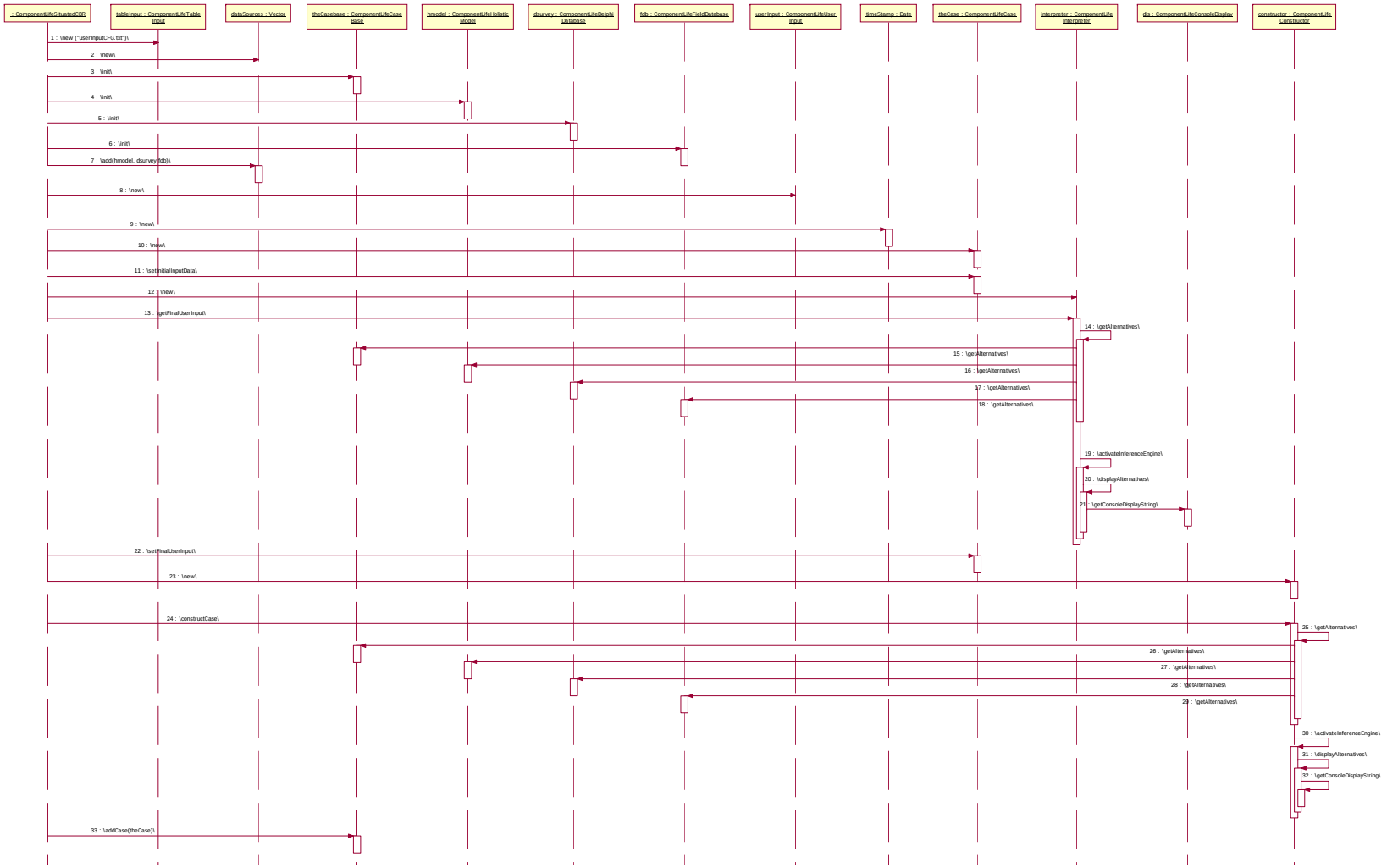


Table 4.4 Extension and modifications points within source codes

Extension / Modification	Related Class.Methods / File	Remarks
Adding new materials to generate more alternatives from the Holistic model.	ComponentLifeHolisticModel()	Add the string representations of these material to the field ComponentLifeHolisticModel.materials
Adding new ways to generate alternatives from the Holistic model.	ComponentLifeHolisticModel() ComponentLifeHolisticModel.getAlternatives(…)	In addition to the ways alternatives are generated by cycling through different materials, additional codes needs to be added to ComponentLifeHolisticModel.getAlternatives(…).
Adding new materials to generate more alternatives from the delphi database.	ComponentLifeDelphiDatabase ()	Add the string representations of these material to the field ComponentLifeDelphiDatabase.materials
Adding new ways to generate alternatives from the delphi database.	ComponentLifeDelphiDatabase () ComponentLifeDelphiDatabase.getAlternatives(…)	In addition to the ways alternatives are generated by cycling through different materials, additional codes needs to be added to ComponentLifeDelphiDatabase.getAlternatives(…).
Adding new materials to generate more alternatives from the field (maintenance) database.	ComponentLifeDelphiDatabase ()	Add the string representations of these material to the field ComponentLifeFieldDatabase.materials
Adding new ways to generate alternatives from the field (maintenance) database.	ComponentLifeFieldDatabase () ComponentLifeFieldDatabase.getAlternatives(…)	In addition to the ways alternatives are generated by cycling through different materials, additional codes needs to be added to ComponentLifeFieldDatabase.getAlternatives(…).
Adding inference engine for interpretation	ComponentLifeInterpreter.activateInferenceEngine(…)	Call out to the inference engine from the body of this function.
Adding inference engine for construction	ComponentLifeConstructor.activateInferenceEngine(…)	Call out to the inference engine from the body of this function.
Modifying tabulated data for similarity index computation	userInputCFG.txt	Change the value of the data under the “Table” heading of the required parameter.
Different ways to compute similarity index	ComponentLifeTableInput	The entire class and all related classes need to be changed.
Using different persistence technologies	ComponentLifeHolisticModel ComponentLifeCaseBase ComponentLifeDelphiDatabase ComponentLifeFieldDatabase	Change all the codes within the bodies of all methods within the wrapper class but maintain their signatures.

4.3 System Testing

The situated CBR framework developed in this project was tested through a series of operation scenarios for the required behaviours as dictated by the specification of the system. It should be noted that the structure of the system is the main focus of this development. The values of the component life calculated in the testing phase of the project do not make any realistic sense.

4.3.1 Implementation Particularities

Initial implementation of the CBR engine had the following points:

- The casebase is implemented by using Java's serialisation mechanisms. All cases are stored in the file: casebase.bin.
- User input is "hard-wired" through a series of codes. Variations in input parameters involve changing the values within the source codes.
- All cases have a fixed computed value of 100.0. This value is "hard-coded" to emulate a result obtained by applying some of the domain heuristics during memory construction. Currently this heuristic is not implemented.
- The values calculated from the system are NOT to be taken as valid as all data used are created artificially to test different paths of the program execution.
- The values obtained from the holistic model, Delphi database, field (maintenance) data are not valid in terms of their reflection of real-life scenarios. Artificial values are created in these systems to populate their contents.

Modifications to allow user input and extraction of real data from the case base and database were subsequently made.

4.3.2 Documentation Scope

Only notes that are relevant to system are listed here, documentation related to unit testing of individual components of the system is not presented. Appendix V outlines a sample of unit testing code for the component ComponentLifeTableInput contained within its main() function. Subsequent developers of the system can run these codes by un-commenting the relevant parts to verify the correct behaviours of the component under consideration.

4.3.3 Output Formatting

An output from a single run of the system is grouped by using the "tab" character. Text strings that relate to the same concept (such as the parameter names and their respective values that describe the user input) are indented from the left margin to

align vertically. Output codes that are colour-coded correspond to colour-coded text found in the test descriptions listed.

4.3.4 Test Scenarios

A series of test scenarios were conducted to test the behaviour of the system through different execution paths. The key behaviours to be tested and their respective test scenarios are listed in Table 4.5.

Table 4.5 Test scenarios and the key behaviours tested

Behaviour to be Tested	Test Scenario
Initial population of casebase	Test Scenario 0: Initialization of Casebase
Retrieval of alternatives from the holistic model, delphi database and field (maintenance) database	Test Scenario 0: Initialization of Casebase
Creation of case	Test Scenario 1: Retrieval from Casebase I Test Scenario 2: Retrieval from Casebase II
Saving of case	Test Scenario 1: Retrieval from Casebase I Test Scenario 2: Retrieval from Casebase II
Retrieval of alternatives from the casebase, holistic model, delphi database and field (maintenance) database	Test Scenario 1: Retrieval from Casebase I Test Scenario 2: Retrieval from Casebase II
Variation in maintenance factor	Test Scenario 3: Retrieval from Casebase III
Variation in maintenance and cleaning factor	Test Scenario 4: Retrieval from Casebase IV
Variation in geographic-location, maintenance and cleaning factor	Test Scenario 5: Retrieval from Casebase V
Variation in geographic-location, maintenance and cleaning factor	Test Scenario 6: Retrieval from Casebase VI
Variation in location-in-building factor	Test Scenario 7: Retrieval from Casebase VII
Variation in geographic-location	Test Scenario 8: Retrieval from Casebase VIII

Test Scenarios 0,1,and 2 must be run consecutively. Test Scenarios 3, 4, 5, and 6 can be run independently. A series of sample runs were conducted with the holistic model, Delphi and field (maintenance) databases. The results from these runs provide the basis for setting the various parameter values in this testing. Further information on testing of the CBR can be found in the Software Testing Report.

4.3.4.1 Computation of Similarity

A key component for producing the required behaviours for the situated CBR system is the computation of the similarity matrix between the sets of user input parameters and the cases contained within the casebase. Figure 4.12 illustrates the key execution paths (as arrows) through different factors and conditions that determine the final similarity index.

Figure 4.12 Key execution paths for computing the similarity matrix

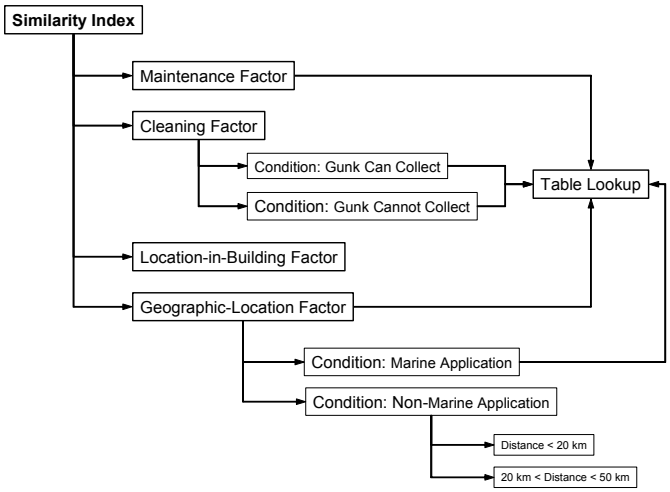
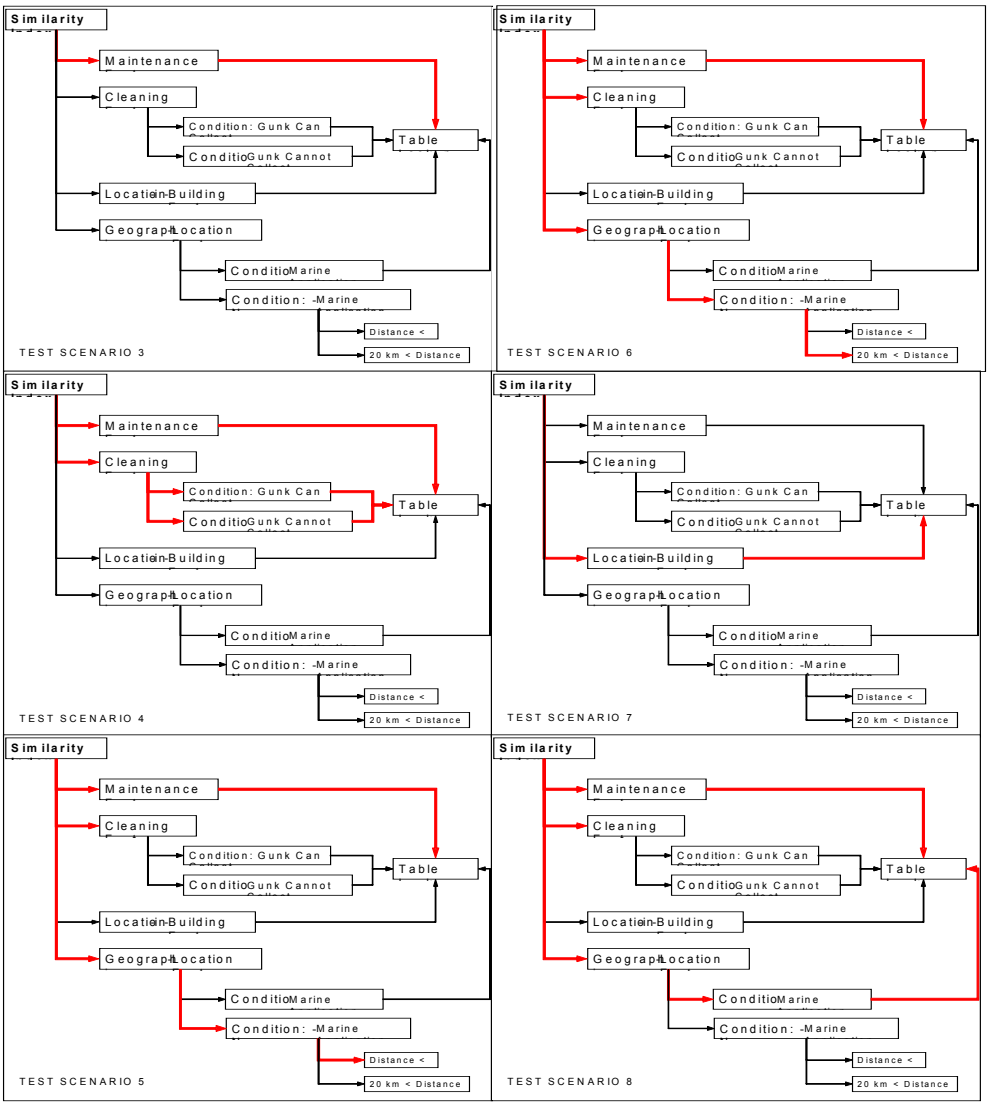


Figure 4.13 Execution paths of different test scenarios



5. QDPW APPLICATION

Lifetime prediction for gutters in Queensland was chosen as an area of concern by the industry partner Queensland Department of Public Works. This would assist in material choice and maintenance scheduling. The general CBR designed for building components was relevant for this, but with some modifications. In particular the holistic model needed modifications to some of its modules (See Figure 3.5) to make it applicable for gutters and the range of materials found for gutters. Those included in the program were:

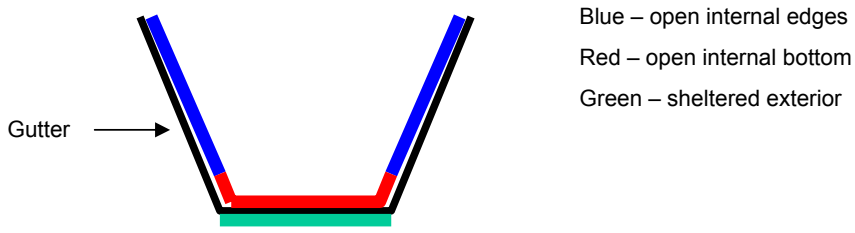
- Galvanised steel
- Zincalume, and
- Colorbond®.

Also the output from the model for metals is a mass loss per year and this needs to be correlated with a predicted life. Consideration also needed to be given to what constitutes the "lifespan" of a product with several criteria for failure being assessed.

5.1 Case Definition for Gutters

Gutters were broken up into different elements or cases as it was considered that the different elements would experience variations in local climate and as such were likely to degrade at different rates. These are shown in Figure 5.1.

Figure 5.1 Representation of the three gutter elements



The bottom of the gutter is the area that will be most affected by an accumulation of dirt and debris, with the internal edges and sides less so. The exterior of the gutter is considered to be 'sheltered' and is not an area where dirt can accumulate. However as a sheltered location it will not be washed by rain and thus marine salt deposited by wind can accumulate.

5.2 Holistic Model Modifications

The holistic model is used to predict the rate of corrosion of metals around Australia. At a particular location the prevailing climatic conditions (of primary importance are the airborne salinity and Time of Wetness -TOW) are used to calculate a mass loss per year. Materials include zinc and steel.

Because gutters are a building component that is classified as a possible dirt accumulation zone, it was considered necessary to formulate new rules for TOW (following wetting events such as rainfall) to be incorporated into the model. Clean, freely-flowing gutters will dry out at a different rate from gutters that have accumulated an amount of leaves and dirt and these rates needed to be determined.

The model also no facility for handling the material of particular relevance to gutters ie Colorbond®. Thus rules for the degradation of Colorbond® had to be devised.

5.2.1 TOW Analysis for Gutters

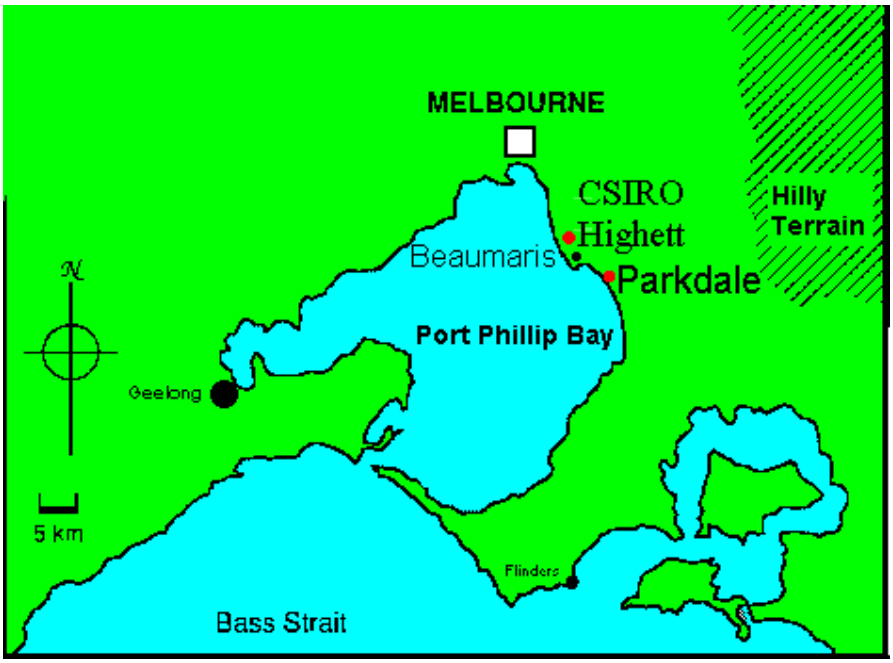
An experiment was undertaken to determine the time a gutter takes to dry after a significant wetting event such as rainfall. Data was needed from both a clean gutter and one which has a build up of dirt and leaf matter. The data collected from this experiment is used for modifying the holistic model as part of the CBR tool. The full experimental details are reported in Report No 2002-059-B No 11. Instrumentation of Roof Gutter to Determine Time of Wetness.

5.2.1.1 Gutter Location

A gutter was selected on site at CMIT, Highett. (Highett is a suburb of Melbourne in Victoria (Figure 5.2

Figure 5.2. It is approximately 3 Km from Port Phillip Bay and has a salt deposition of approximately 8 mg/m²•day and a corrosion rate for steel of approximately 10µm/year.)

Figure 5.2 Location of Highett, a suburb of Melbourne



The CSIRO site at Highett also has an exposure station which is well characterised in terms of weather, corrosion and salt deposition. The exposure station is at the north end of the site and the building used is approximately 50m to the east of the Weather station.

The building used for instrumentation of the gutter (shown in Figure 5.3) has an asbestos roof and a galvanised gutter. The gutter has both clean sections and sections that have a significant build up of leaf litter and dirt. The gutter has red rust in the sections that are dirty and some coating loss from the clean sections. The gutter runs in a north south direction and the roof and gutter do not have any trees directly overhead.

Figure 5.3 View of building used for instrumentation of the gutter

5.2.1.2 Location of Sensors in Gutter

Figure 5.4, shows the inside of the gutter from the south. It can be seen that the southern end has significant build up of leaf litter and dirt. The dirt and leaf litter is approximately 10mm deep.

Figure 5.4 View of the inside of the gutter

In the clean section of gutter a wetness sensor and surface temperature sensor were mounted, while in the dirty section only a wetness sensor was installed. Figure 5.5 (a) shows the clean section of gutter and the sensors attached to the bottom with thermally conductive tape. The clean section has some surface dirt but is generally clean.

Figure 5.5 Placement of sensors in (a) clean and (b) dirty section of gutter

(a) (b)

Figure 5.5 (b) shows the dirty section and the location of the wetness sensor. The dirt and leaf litter was carefully lifted and the bottom of the gutter cleaned, before the sensor was stuck to the bottom of the gutter and the leaf litter and dirt replaced.

5.2.1.3 Data Collected

Data was collected from the gutter for the period from the 9th of February to the 29th of March 2005, at 15 minute intervals. This data from the gutter was combined with data from the Weather Station. Figure 5.6 shows a small section of the data collected.

The graph shows that a wetness event, rain, occurred on the morning of the 15th of February and that both the clean and dirty sections of the gutter became wet. The lag between the clean and the dirty wetness sensor is because the dirt and leaf litter in the dirty section of the gutter need to wet before the gutter and sensor become wet. For these experiments, the start of the drying period is timed from when the weather station wetness sensor starts to dry. The drying period is considered ended when the gutter wetness sensor has returned to zero.

Table 5.1 details drying times for the gutter from the data collected. For the graph in Figure 5.6 the drying time for the clean gutter was only 1½ hours while the dirty gutter took over 73 hours.

Figure 5.6 Graph of data from 15 - 19 February 2005

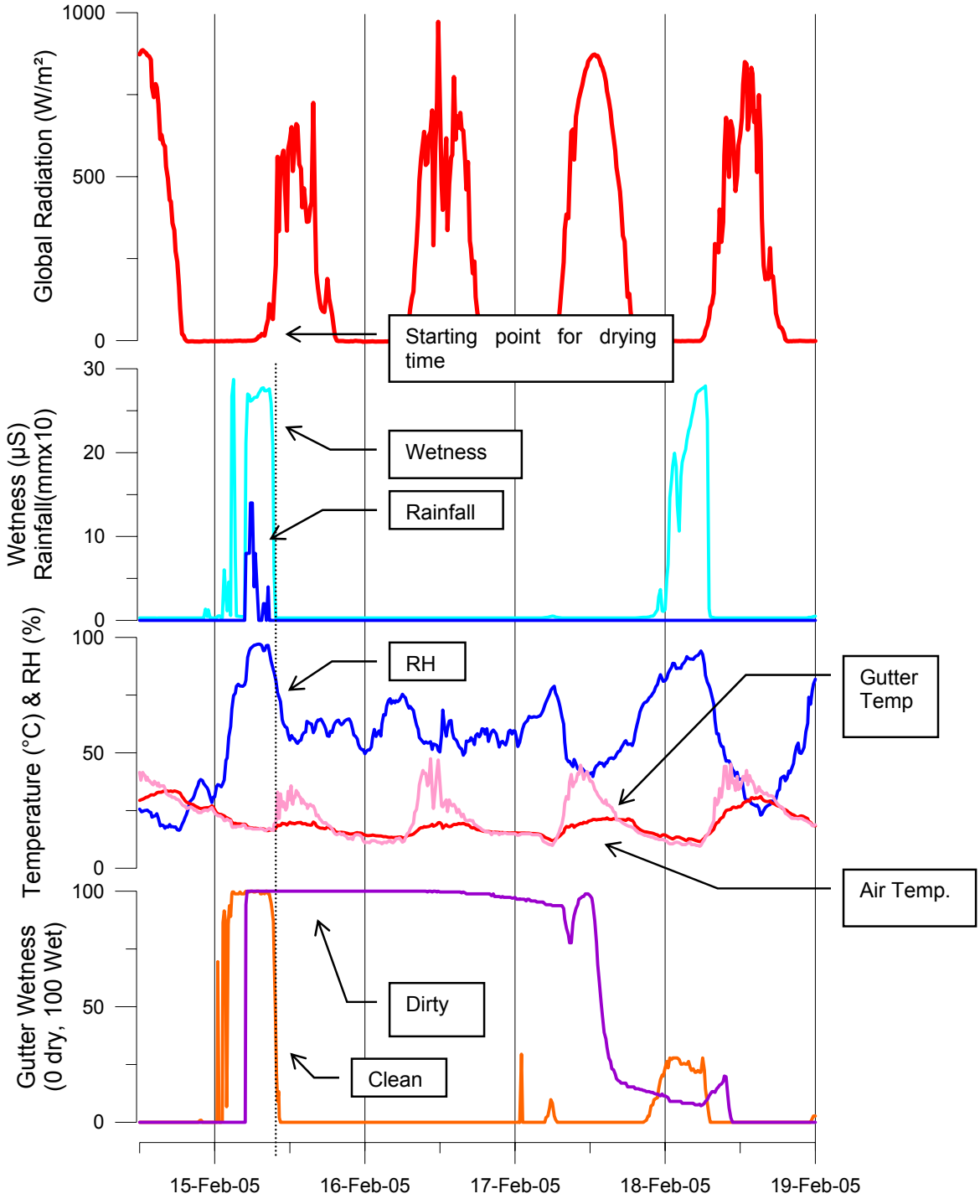


Table 5.1 Drying times for the gutter after significant wetting events

Dirty Gutter		
Time of Wetness event	**Time of Dry Gutter**	**Time (hours: minutes)**
12/02/2005 05:15	14/02/2005 11:15	54:00
15/02/2005 09:00	18/02/2005 10:45	73:45
08/03/2005 08:00	09/03/2005 19:30	35:30
11/03/2005 08:00	13/03/2005 06:00	46:00
	Average	**52:20**
Time of Wetness event	**Time of Dry Gutter**	**Time (hours: minutes)**
12/02/2005 05:15	12/02/2005 08:40	3:30
15/02/2005 09:00	15/02/2005 10:20	1:30
02/03/2005 06:45	02/03/2005 08:45	2:00
06/03/2005 07:15	06/03/2005 09:00	1:45
08/03/2005 08:00	08/03/2005 10:15	2:15
10/03/2005 08:00	10/03/2005 10:45	2:45
	Average	**2:15**

Drying times were only calculated after significant wetness events occurred, ie. either rain or when the wetness sensors, both gutter and air, reached their maximum value. The times calculated here have been used in the holistic model as indicative of the drying times of clean and dirty gutters. However some variation will occur due to a whole range of circumstances eg. variation in relative humidity and temperature (measurements were taken in Melbourne late summer), and the extent of sheltering of the gutter.

Table 5.2 shows a summary of variables typically used for corrosion studies. TOW is the time of wetness expressed as a percentage of time, the ISO TOW is when the relative humidity is greater than 80% and temperature is greater than 0°C according to ISO 9223. The Gutter TOWs are based on the wetness sensors and the Air TOW is the wetness sensor on the weather station.

Table 5.2 Summary data

Variable	**Value**	**Units**
ISO TOW	22.8	% of time
Gutter TOW Clean	54.0	% of time
Gutter TOW Dirty	37.2	% of time
Air TOW	14.9	% of time
Average Air Temp	18.0	°C
Average Gutter Temp	20.8	°C
Average Air RH	65.9	%

It is interesting to note that the clean gutter has a longer TOW overall (54%) than the dirty section (37%). This is because the clean section is wet nearly every night due to condensation events while the dirt and leaf litter in the dirty section absorb a certain amount of water before the gutter or sensor get wet.

5.2.2 Theoretical Analysis of Gutter Drying

5.2.2.1 *Evaporation from the gutters.*

A theoretical analysis of gutter drying was carried out with the key points highlighted below.

As a rough guide, water 2.5 mm deep evaporates in about 1 hr at 50% relative humidity in moving air (nominally 2 m/s along the gutter). The nominal 2 m/s is estimated from a wind averaging about 5 m/s at eaves height; results are not very sensitive to the actual wind velocity unless the nominal velocity is less than 0.5 m/s.

For a new gutter with correct slope the water left after rain is of the order of 0.2 mm thick and evaporates at 75% relative humidity (2 m/s air) in about 10 minutes.

For an old gutter that has sagged and filled part way up with gunk the equivalent water depth may be 2 cm thick and require 16 hours to evaporate at 75% relative humidity (2 m/s air).

Evaporation of water trapped in a 2.5 cm thick layer of gunk is expected to proceed at about the same rate as a 2.5 cm thick layer of water. The initial evaporation rate is the same, and as it dries up ‘wicking’ action of the gunk brings water up from below to keep the top wet where evaporation is taking place.

5.2.2.2 Statistical effects on evaporation.

Both relative humidity and wind speed are statistical effects. Statistically rare events can have a significant overall effect. In about 7% of cases the wind speed is very low or the relative humidity is very high. In those cases the evaporation time is longer by about a factor of 13, giving 8 ½ days for a gunk filled gutter.

5.2.3 Colorbond® Degradation Model

Essentially, there are six gutter types in Australia:

1. Galvanised steel
2. Painted galvanised steel
3. Zincalume coated steel

4. Painted zincalume coated steel
5. Colorbond® with one-sided topcoat
6. Colorbond® with two-sided topcoat

Modeling the degradation of these six gutter types is approached as follows.

1. Galvanised steel. The degradation of galvanised steel products is predicted directly from the current holistic model for galvanised materials.

2. Painted galvanised steel. Here the application of paints to the gutter is carried out after installation. Quality control on such paint films is poor and a range of different paint formulations may be used. The use of corrosion inhibited primers is not a formality in such systems and lifetime predictions are essentially meaningless. Depending upon the location of the gutter, any standard paint coating on galvanised steel will offer limited protection to the gutter over time periods exceeding five years (Sjöström, 1990).

3. Zincalume coated steel. The degradation of zincalume coated steel products is predicted directly from the current holistic model for zincalume materials.

4. Painted zincalume coated steel. Modeling is not meaningful. See above for explanation for 2. painted galvanised steel.

5. Colorbond® with one-sided topcoat. Colorbond® is a product of Bluescope steel and has been proven to have exceptional performance in most locations across Australia. Although there are different grades of Colorbond®, the most common make-up for guttering is steel sheet (low carbon steel) with a coating of zincalume AZ 150 (150 g m^{-2}), which is overcoated on both sides with a 5 μm chromate-containing epoxy primer. The one-sided product has a 20 μm thick UV-resistant topcoat and a 5 μm grey backing coat covering the primer (Bluescope Steel, 2005). Colorbond® gutters are assembled so that the backing coat forms the interior of the gutter and the coloured topcoat forms the outer gutter.

Inspections have found that before Colorbond® is installed the 10 μm backing coat is riddled with holes, whilst the topcoat shows few defects. An inspection of a length of Box-type gutter using a holiday tester showed the backing coat to possess approximately 1000 point defects per square metre. Inspection of ten panels (0.6 m × 0.6 m) of Colorbond® topcoat with a holiday tester estimated damage to be limited to approximately 2 point defects per square metre. The number of defects was also increased at folded edges. Holiday testing of fold lines on the backing coat of the Box-type gutter showed almost constant breaks in the coating. The folded edges on the topcoat were not initially damaged.

Most paint films are thought to be best modelled with either a localised mechanism only or a combination of localised and general mechanisms (Sjöström, 1990). Colorbond®, which comprises of two organic layers, a thin epoxy primer containing

inhibiting pigments and a poly vinylidene fluoride topcoat has been shown to fail in localised areas. Inspection of the defects located on unexposed Colorbond® suggested that they had a diameter in the range of 50 µm. The model presented here assumes the following:

- Colorbond® is installed using best practice: that fasteners and washers are selected as per the manufacturers specifications, that materials are not damaged or exposed to contamination sources, and that neutral-cure silicone sealants are used for edge protection.
- It is assumed that the Colorbond® topcoat has isolated defects with a 50 µm diameter. The backing coat is assumed to have 50 µm defects that are interconnected with other defects, thus giving rise to an increased rate of chromate leaching from the primer.
- That strontium chromate is leached from the 5 µm thick primer layer and is able to inhibit corrosion until it is depleted through either washing or precipitation into a more insoluble form (Furman et al., 2005; Wang et al., 2004; Scholes et al., 2005; Sinko, 2001; Zin et al., 1998).
- The presence of salts enables moisture to be condensed at relative humidities below either 35% RH (sea salt) or 75% RH (sodium chloride) (Muster and Cole, 2005). The presence of salts encourages chromate leaching to occur at an increased rate (Prozek and Thierry, 2004). The concentration of salts on the surface and the surface condition (dry, wet) is calculated using the existing holistic model.
- As soluble chromate levels are depleted corrosion commences on the zincalume layer (Baghni et al., 2004). Corrosion rates are assumed to be directly proportional to the concentration of salts on the surface (Davis et al., 1987).
- The corrosion of zincalume is assumed to occur with an aspect ratio of 50:1. That is, for every micron in depth of penetration into the coating, the coating is undercut 50 microns either side of the defect. At total penetration of the 20 µm thick zincalume it is therefore assumed that an area 2 mm in diameter is being attacked by the environment. The aspect ratio was developed from cross-sections of Colorbond® materials where damage had occurred.
- Zincalume corrosion within the defect is assumed to occur at a rate similar to that of an uncoated surface. Therate of zincalume mass loss for a given salt deposition rate over a one year period is used (King et al., 2001; Ganther and Cole, 2002).
- Once the depth of corrosion damage reaches 20 µm into the zincalume corrosion rate is accelerated by the exposure of steel according to relative steel:zincalume areas available for galvanic corrosion (Bluescope Steel, 2005; Tada et al., 2004).
- Once a steel area of 0.25 cm radius is exposed, the zincalume is assumed not to galvanically protect the underlying steel any more and steel corrosion occurs (Tada et al., 2004).
- Mild steel corrosion within the defect is assumed to occur at a rate similar to that of an uncoated surface. The rate of steel mass loss for a given salt deposition rate over a one year period is used (King et al., 2001).

6. Colorbond® with two-sided topcoat. Will be dealt with as detailed above for Colorbond® with topcoat and backing sheet but with two topcoats.

5.2.3.1 Modelling: Equations and interdependencies

Figure 5.7 provides a visual representation of the sequence of steps used to model Colorbond® degradation.

Figure 5.8 provides a schematic of the interdependencies for the modelling of metal degradation under an organic coating consisting of primer and topcoat. The presence of chromate pigments enable corrosion to be limited when in the presence of water and salts. Once chromate is leached, either directly from the primer or through the topcoat or backing coat, corrosion of the zincalume is initiated. Chromate leaching is enhanced by water, salts and UV exposure. Corrosion is accelerated by the presence of water and salts. In reality, the onset of zincalume corrosion beneath the paint film is likely to create an increased defect volume and also, in some instances, expose the defect to fresh reserves of chromate. The current model does not consider the availability of new reservoirs of chromate.

5.2.3.2 Modelling Chromate depletion

Chromate leaching from an epoxy-polyimide polymer has been shown to yield 200 µg per square centimetre of exposed primer during a 10-day immersion (Scholes et al., 2005). Chromate is assumed to leach according to Fick's second law (i.e. $t^{0.5}$ dependence). There is some conjecture as to whether leaching is likely to be Fickian (Furman et al., 2005). However, the $t^{0.5}$ dependence is likely to provide a reasonable fit for "universal" primer systems. Work by Zin et al. (1998) and Sinko (2001) observed a $t^{0.5}$ dependence.

The leaching rates used for the model are derived from the 10-day immersion data of Scholes et al. (2005), which is modelled with a $t^{0.5}$ relationship as shown by the thin solid line in Figure 3. Additional leaching of primer through the back coat and top coat of Colorbond® is assumed to follow an identical $t^{0.5}$ relationship but is assumed to be limited by an increased barrier to diffusion which is calculated directly from the thickness of the backing coat and top coat (see dashed lines in Figure 5.9).

Figure 5.7 **Model for the degradation of Colorbond® materials. (a) A 50 mm diameter defect in the organic coating is assumed, (b) chromate is leached from the primer due to the presence of moisture and salts, (c) upon depletion of chromate inhibitor zincalume is corroded with an aspect ratio of a/d = 50, (d) where d exceeds the thickness of zincalume, surrounding zincalume is lost at an increased rate due to galvanic corrosion. Steel corrosion is assumed to occur when g > 1 cm and zincalume no longer provides sufficient galvanic protection for the underlying steel.**

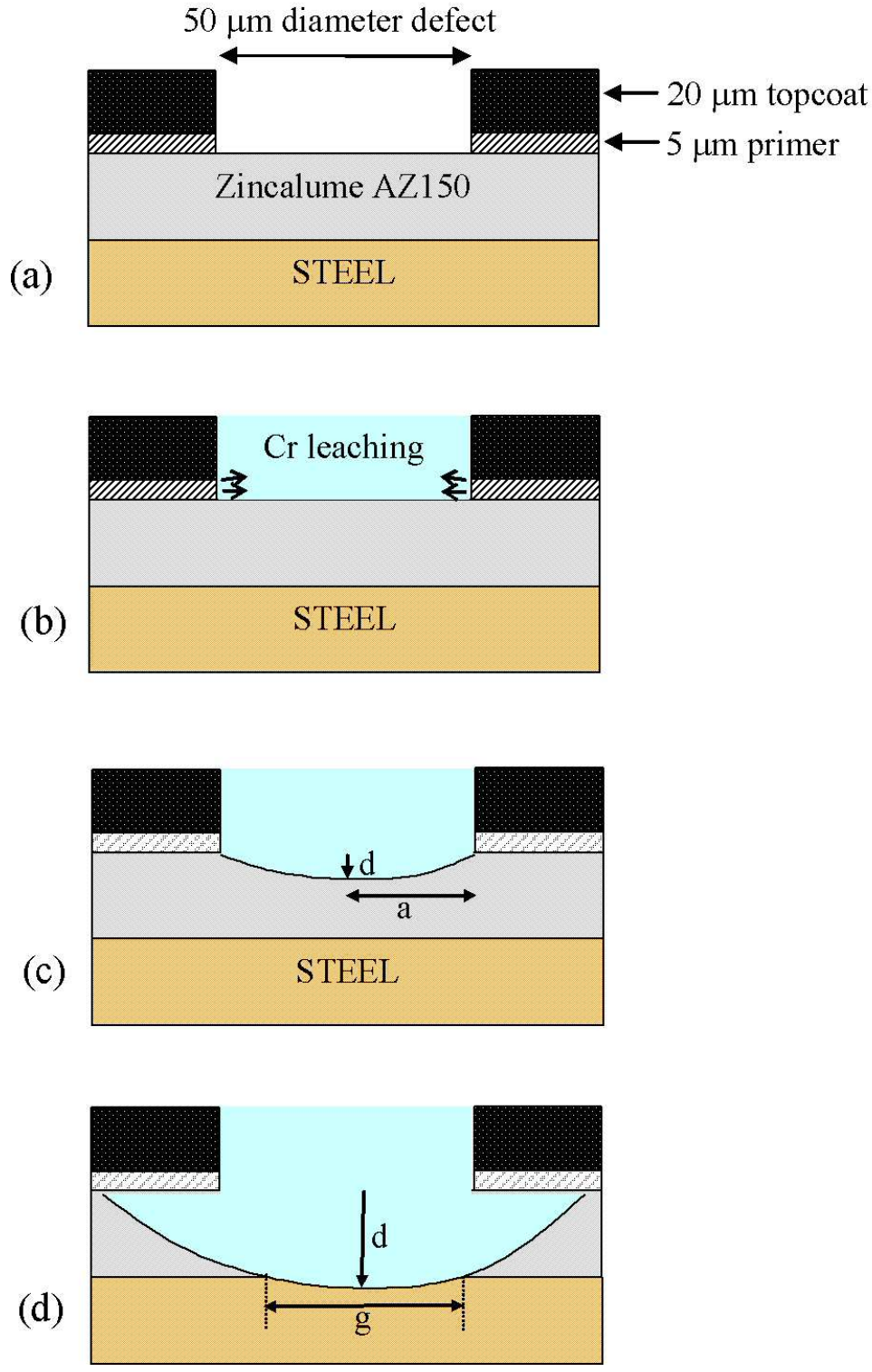


Figure 5.8 Circular relationships determining metal degradation at a defect in a primer and topcoat.

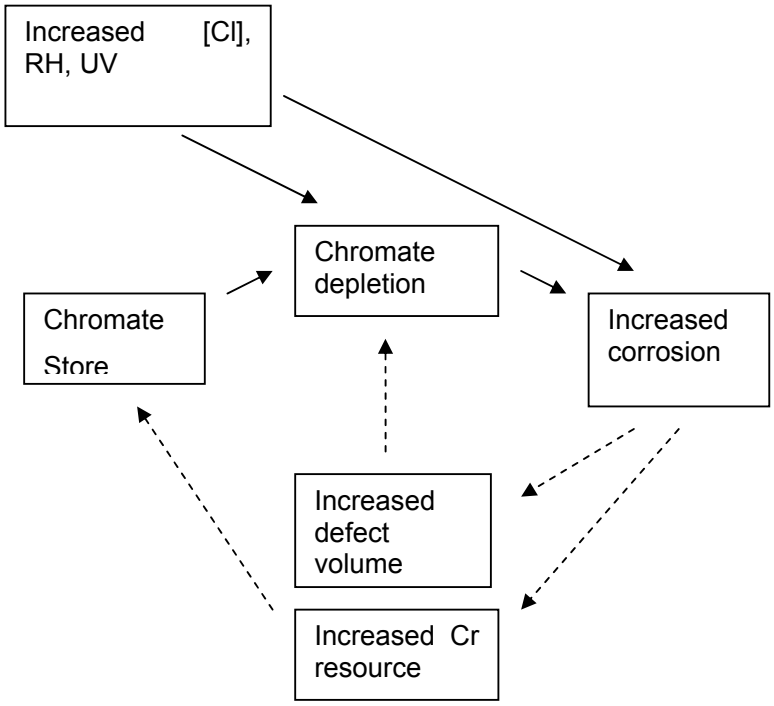
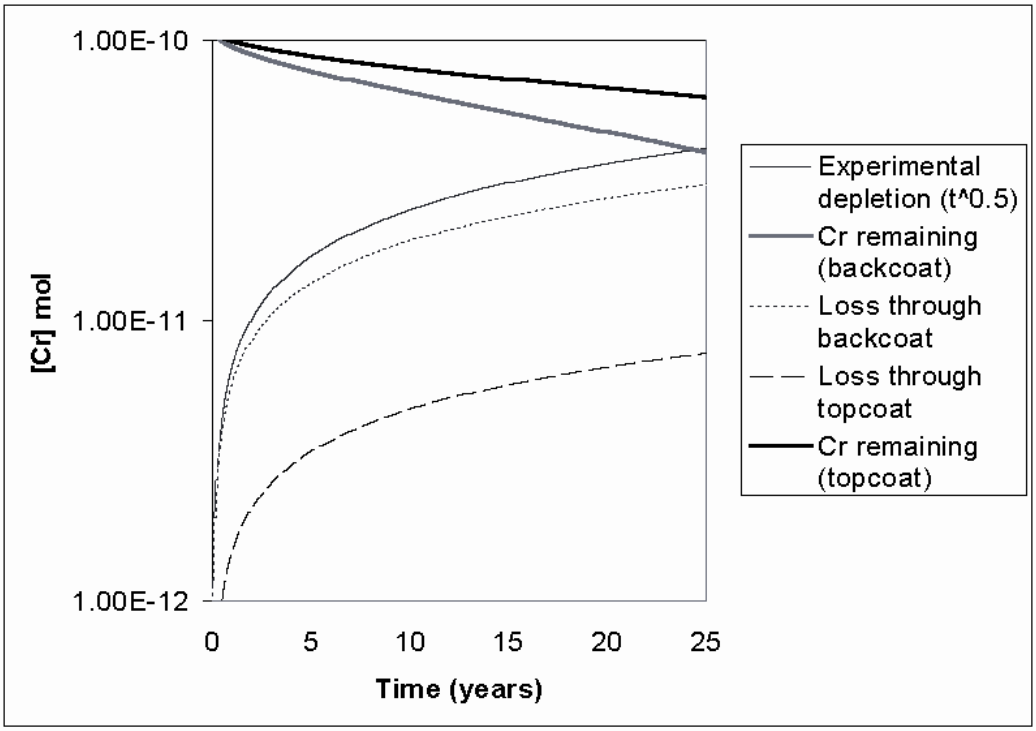


Figure 5.9 Chromate depletion model. Depletion occurs directly from primer in defect and through backing coat and to a lesser extent through the topcoat.



5.2.3.3 Influence of salt concentration on leaching rate

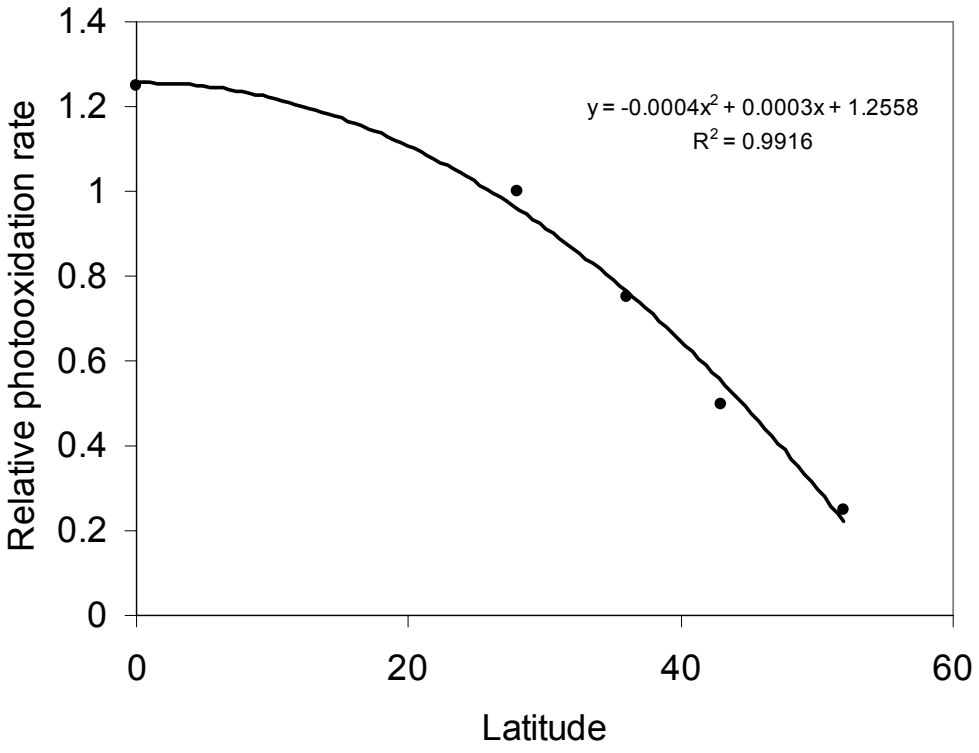
Chromate leaching has also been shown to be a function of chloride concentration (Prosek and Thierry, 2004). Chloride anions are able to associate with soluble chromate and encourage dissolution of pigments. Prosek and Thierry (2004) found that 10 mmol L^{-1} of NaCl increased chromate leaching by 30%. By increasing the NaCl concentration to 100 mmol L^{-1} had minimal additional impact. The amount of chromate leached during a 3-hour period allowing for the influence of salt concentration is given as:

$$L_{Cl} = (L_t - L_{t-1}) * (1.2123[Cl]^{0.1544}) \quad \text{…Eqn 5.1}$$

5.2.3.4 Influence of photooxidation on leaching rate

Data from Bauer (2000) shows that the relative photooxidation rate can be correlated to latitude coordinates for the northern hemisphere. Here, it is assumed that the correlations hold for the southern hemisphere. The data presented in Figure 5.10 allows for time of UV exposure, UV spectrum changes and for the influence of relative humidity.

Figure 5.10 Dependence of photooxidation rate on latitude. Data from Bauer (2000)



Correlation of the relative photooxidation rates with damage to Colorbond® was achieved by matching the % failure data provided by Bauer (2000) with the failure ratings of Colorbond® provided by King et al. (2001). Failure of the topcoat and backing coats will lead to increased loss of chromate from the primer. The total loss of chromate is given as:

$$L_{total} = L_{Cl}(1+xt)(-0.0004*\mathrm{LAT}^2 + 0.0003.\mathrm{LAT}) \quad \text{mol} \qquad \text{…Eqn 5.2}$$

where x = 0.8 for topcoat and 0.4 for backing coat, t = time in years, LAT = latitude in degrees.

The amount of chromate remaining in the 25 μm area surrounding the defect is given as:

$$Cr_{rem} = 1.084 \times 10^{-10} - L_{total} \text{ mol} \qquad \text{…Eqn 5.3}$$

5.2.3.5 Protection by chromate

The amount of protection offered against corrosion by chromate is calculated using a dependence factor, Cr_{dep}, and is obtained from electrochemical data (see Figure 5.11) that defines corrosion rates (currents) as a function of available chromate concentration [Cr].

$$Cr_{dep} = \frac{0.15 + 1.85e^{-[Cr]/0.00002}}{0.15} \qquad \text{…Eqn 5.4}$$

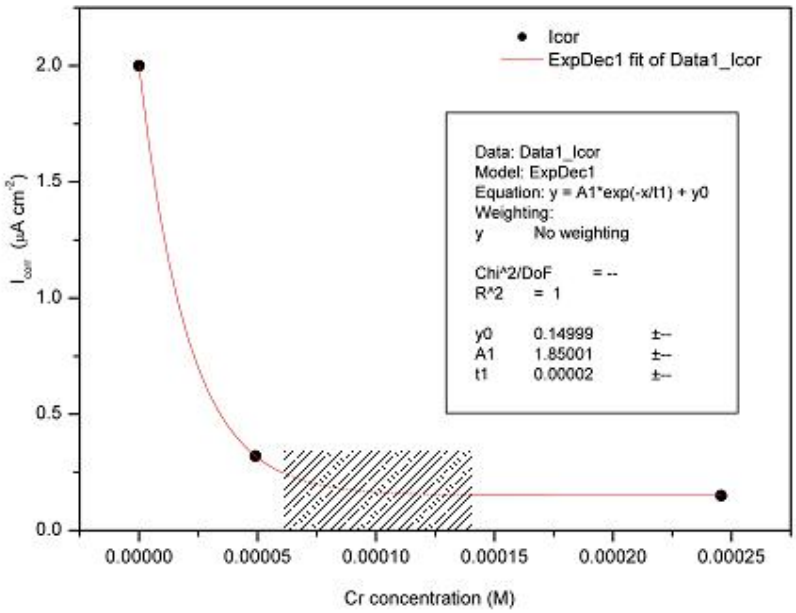
Cr_{dep} has a maximum value of 2. The available chromate concentration [Cr] is derived from equation 5 and is a function of the recently chromate released [$L_{total}(t)$-$L_{total}(t-1)$] into the volume of the defect and the fraction of chromium remaining in the primer surround the defect.

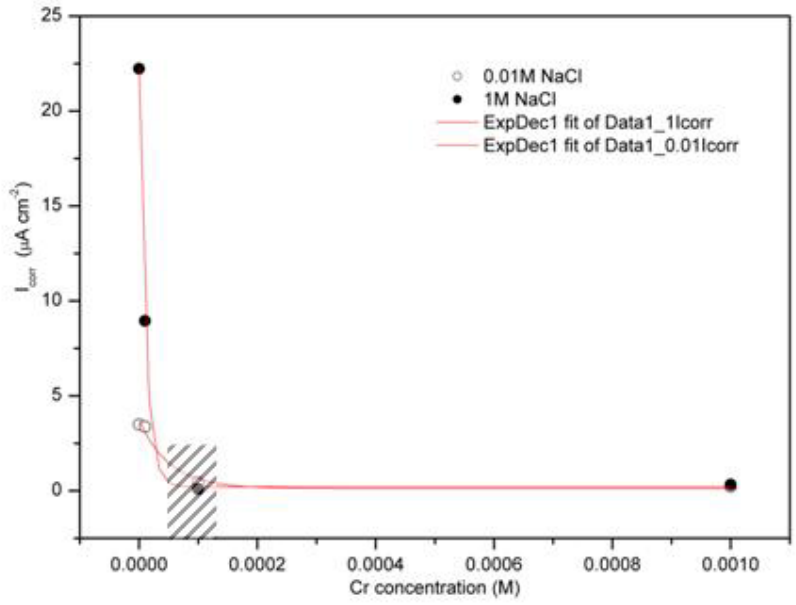
$$[Cr] = \frac{[L_{total}(t) - L_{total}(t-1)]}{V} \frac{Cr_{rem}}{1.084 \times 10^{-10}} \qquad \text{…Eqn 5.5}$$

The volume, V is the volume of the defect plus the 25 μm of the primer surrounding the defect.

$V = \pi(50)^2 5$ μm = 3.92699×10^{-11} L

Figure 5.11 **The influence of chromate concentration on the corrosion current of zincalume at varying chloride concentration. Shaded area represents the typical concentrations of chromate during leaching.**





5.2.3.6 Zincalume mass loss

The actual corrosion is given by:

$$d_{ZA} = M_{ZA} \times (Cr_{dep} - 1) \qquad \text{…Eqn 5.6}$$

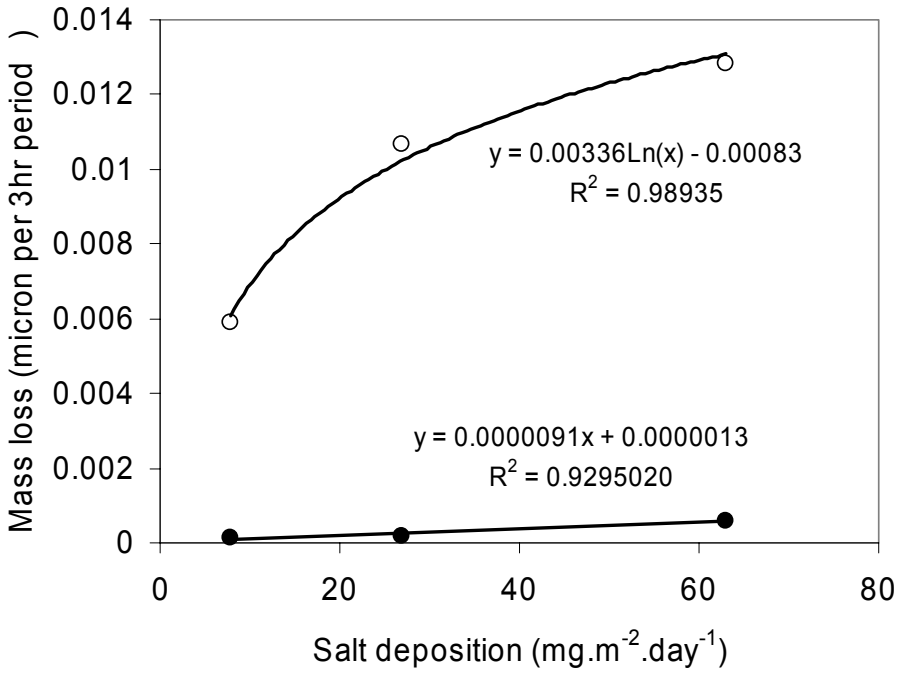
and is maximum when $Cr_{dep} = 2$, which occurs when [Cr] < 5×10^{-5} mol L^{-1}. Under the set conditions when [Cr] exceeds 2.9×10^{-4} mol L^{-1} no corrosion occurs. These values are in affect generated from the experimental data presented in Figure 6. Values for M_{ZA} and M_{STEEL} are calculated from the work of King et al. (2001), who quoted the yearly corrosion rates given in Table 5.3.

Table 5.3 Corrosion rate data from King et al. (2001).

Site	Rain days/yr	Estimated time-of-wetness (% > 75)	Salt ($mg/m^2.day$)	Zincalume 1yr (μm/yr)	Mild steel 1 yr (μm/yr)
Navy (Flinders)	161	80	63	1.413	30
Waterboard		70	27	0.343	21.8
CSIRO		65	7.8	0.237	11.2

Values of corrosion rates are modified by the estimated time-of-wetness to allow a 3-hourly corrosion rate to be calculated. Figure 5.12 shows the generation of corrosion rates as a function of salt deposition allowing for time-of-wetness.

Figure 5.12 Correlations between mass loss and average salt deposition rate for three sites. Data from King et al. (2001).



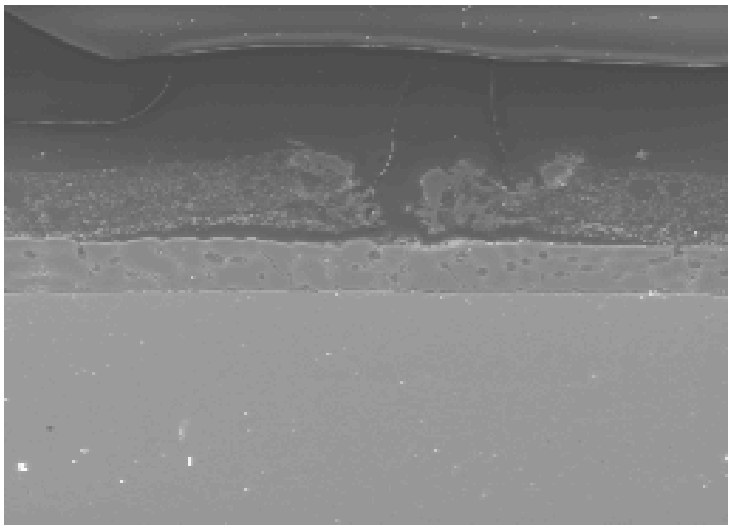
Once the zincalume mass loss reaches 20 μm the underlying steel is exposed. At this point the exposed steel is able to fill the role as a cathode and promote zincalume corrosion at an increased rate. Electrochemical testing revealed that zincalume corrosion currents are approximately 1.42 times greater when coupled to an equal area of mild steel, where the separation distance was less than 1 cm. For this reason when d > 20 μm, d_{ZA} is multiplied by a factor of 1.42.

5.2.3.7 Steel mass loss

Steel mass loss is not expected to occur until zincalume surrounding the area is consumed. Zincalume mass loss is assumed to occur with the shape of a spherical cap (see Figure 5.7) where d/a = 1/50. The d/a ratio was adopted from cross-sectional analysis of a damaged area after significant damage (Figure 5.13). Once a steel area of 0.5 cm radius is exposed, the zincalume is assumed not to galvanically protect the underlying steel any more and steel corrosion occurs (Tada et al., 2004). The 0.25 cm radius area occurs when the effective d_{ZA} > 50 μm. Steel corrosion for a three hour period is given by:

$$M_{STEEL} = 0.00336 \ln[Cl] - 0.00083 \,\mu\text{m} \qquad \text{…Eqn 5.7}$$

Figure 5.13 SEM cross-section showing typical damage at the site of a defect. Sample shown was exposed to 35 cycles of GM9540P accelerated corrosion test.



A summary of the inputs for the model are included in Table 5.4.

Table 5.4 Inputs, parameters and details of mathematics within the Colorbond® degradation model.

Parameter	Symbol	Units	Value	Description
Surface condition	S		0 = dry 1 = condensed moisture 2 = raining	Determines whether wet or dry. Derived from relative humidity and surface temperature data in holistic model.
Cumulative time-of-wetness	TOW_{cum}	hours	When S = 1 or 2, $=10^{\log(L_{cum})-0.5(\log\frac{TOW_{cum}}{TOW_{cum}+3})}$ For both topcoat and backing coat: L_{cum}=1.41E-13 mol when TOW_{cum}= 0 hrs.	Cumulative time-of-wetness, where S = 1 or 2 allows leaching of Cr from primer according to Fick's second Law.
Leached Cr	L_{cum}	mol	L_{cum} is the running accumulation of chromate that would be leached in the absence of chloride anions.	L_{cum} is dependent upon the area of primer exposed and not on the total chromate concentration or liquid volume.
Additional leaching	L_{add}	mol	Back: $=10^{\log(L_{cum})-0.5(\log\frac{TOW_{cum}}{TOW_{cum}+3})}$ Top: $=10^{\log(L_{cum})-0.5(\log\frac{TOW_{cum}}{TOW_{cum}+3})}$	The additional leaching of Cr through the backing coat or topcoat. Initial values, backcoat =1.128E-13, topcoat = 2.282E-14.
Salt modified leach rate	L_{Cl}	mol	= $((L_{cum}+L_{add})*1.2123*[Cl]^{0.1544})$	The loss of chromate in a single 3-hour period given that a certain concentration of chloride is present on the surface.
Latitude	LAT	Degrees		Input to describe the likely photooxidation rate (UV) exposure of paint films.
Sun/Salt leaching	L_{total}	mol	=(1+x*time)*(-0.0004*LAT^2+0.0003*LAT+1.2558)*L_{Cl}	Leaching as a result of both sun and salt.
			x = 0.8 for topcoat, 0.4 for backing sheet.	x values are derived from Bauer (2000).
Cr remaining	Cr_{rem}	mol	Initial - L_{total} Cr_{rem} = $1.084 \times 10^{-10} - L_{total}$ mol	Initial Cr present minus the cumulative sum of all leached chromate. Assuming the primer contains 20 % v/v strontium chromate, the total available pigment in a 25 μm zone surrounding the defect is of the order of 0.20×18.4 mmol $cm^{-3} \times 2.945 \times 10^{-8}$ cm^3 = 1.084×10^{-10} mol. Cr has been shown to leach from no further into epoxy-based paints than about 25 μm from a defect.

Parameter	Symbol	Units	Value	Description
Defect volume	V	L	= 3.92699E-11 for both topcoat and backing coat.	Initial volume (50 μm damage) + 25 μm area surrounding damage. 25 μm surrounding damage is the accessible area for Cr to leach from. Volume is generated from the 5 μm nominal thickness of primer.
Salt concentration	[Cl]	mg/m^2.day	Cumulative salt deposition derived from holistic model.	
Active chromate concentration	[Cr]	mol/L	$[Cr] = \frac{[L_{total}(t) - L_{total}(t-1)]}{V} \frac{Cr_{rem}}{1.084 \times 10^{-10}}$	The estimated amount of chromate in mol/L available to prevent corrosion. Calculated based upon leached amount and volume.
Non-Cr zincalume mass loss	M_{ZA}	micron	Where d > 20 μm, M is multiplied by 1.42 due to increased galvanic corrosion resulting from steel exposure. =0.0000091+0.0000013[Cl]	Mass loss of zincalume calculated based upon holistic model at a given salt accumulation.
Chromate dependence	Cr_{dep}		(0.15+1.85*EXP(-Cr_{dep}/0.00002))/0.15	Dependence of corrosion rate on chromate concentration
Actual zincalume mass loss	d_{ZA}	micron	(Cr_{dep}-1)*M_{ZA}	Estimated real damage to zincalume in terms of depth.
Cumulative actual zincalume mass loss	d	micron	$\sum d_{za}$	Sum of corrosion damage
Steel corrosion	d_{STEEL}	micron	0.00336*LN[Cl]-0.00083	Where g > 50, d is predicted by holistic model for steel mass loss given a certain salt accumulation.
Cumulative metal mass loss	d_{total}	micron	$= \sum d_{za} + \sum d_{st}$	Total depth of penetration into substrate.

5.2.3.8 Preliminary data from model

As a demonstration, the model predictions for the locations of Flinders Naval Base (Victoria) and Brisbane, Cairns and Charleville (Queensland) are presented. Details of the input parameter values used to generate results are provided in Table 5.5.

Table 5.5 Abbreviated meteorological data and estimated salt deposition rates.

Location	Latitude	Estimated % time where water is condensed on surface or rain event.	Estimated salt deposition (mg/m^2.day)
Flinders Naval Base	38.3	90 %	300
Brisbane	27.5	50 %	20
Cairns	16.9	83 %	15
Charleville	26.4	33 %	5

Figure 5.14 provides graphical calculations for the model based upon the inputs from Table 5.5. Further qualification of the model linked to the holistic model is required to confirm the accuracy of this first attempt model for Colorbond degradation. The suggested outputs from the model are:

- time for chromate depletion (time to white rusting, d > 0)
- time to penetration of zincalume (d = 20 μm)
- time to red rust initiation (d = 50 μm)
- time to hole generation through sheet (d > 500 μm) [Steel ~ 600 μm thick].

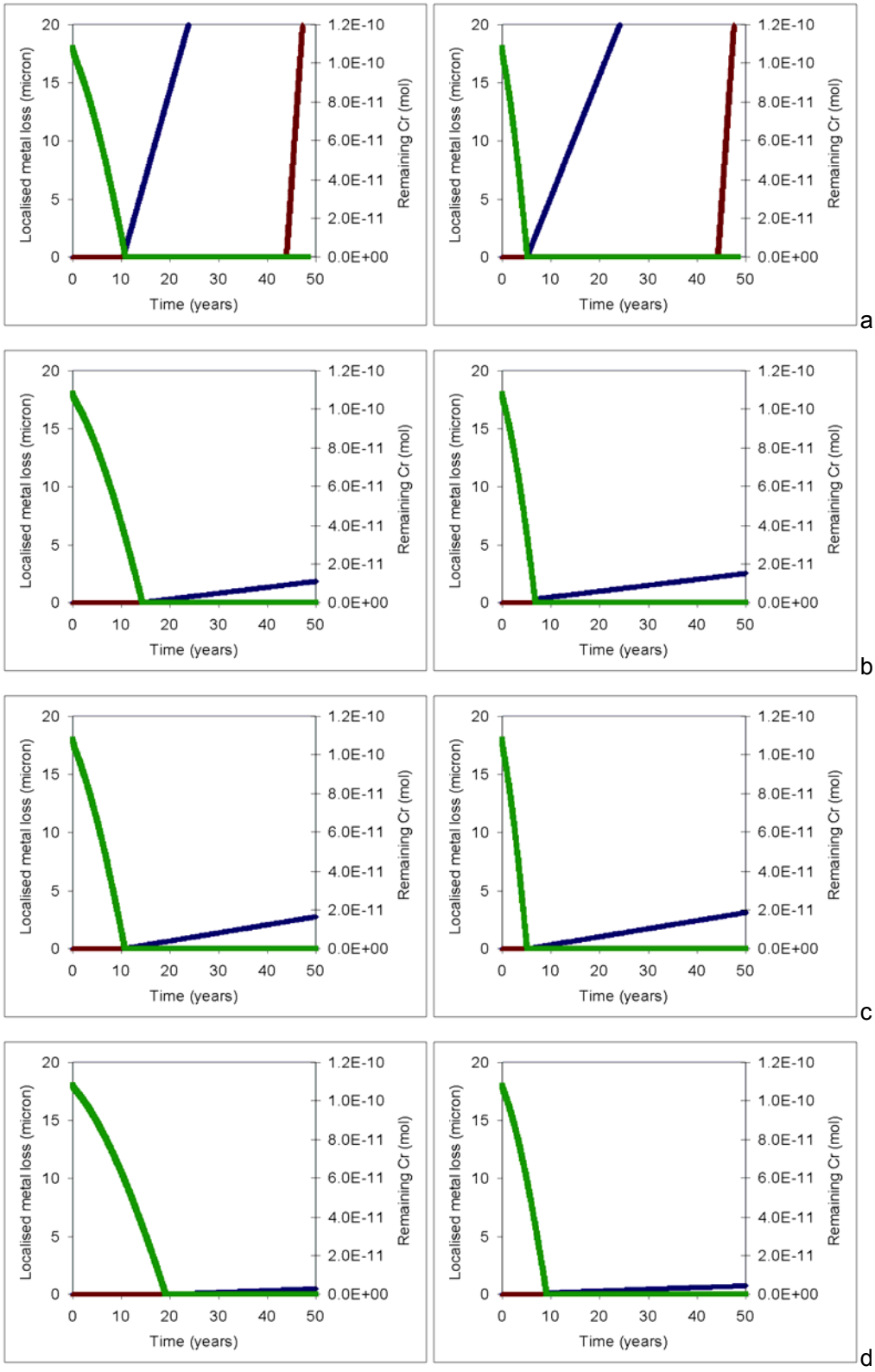
5.2.3.9 Discussion of Colorbond® model

The model produced to date is essentially an attempt to model the degradation of an extremely complex system. Colorbond®, or painted items in general, are difficult to quantify due to the often unpredictable nature of installation and treatment of materials. For instance, a scratch in the paint or incorrect protection at the edges will have a much larger influence on the longevity of a painted article than a 50 μm defect assumed in the current model.

Major errors and complications of the model include:

- The exact geometry of the defect is not likely to be 50 μm and geometry of zincalume corrosion and paint delamination is likely to vary in many cases.
- The wetting and drying rates within a defect and under the paint film are likely to vary from the surface as a whole.
- Factors affecting or influenced by adhesion of the paint are not considered.
- Salt concentrations within the defect are likely to be higher than on the open surface. This is expected to result in a more rapid corrosion of the underlying metal. No factor is currently incorporated to allow for this affect. However, straightforward experimental studies could validate such factors.
- The exact dimensions of the zincalume and paint films are likely to vary and also the amount of available chromate pigments in a localised area will vary.

Figure 5.14 Output data from Colorbond® degradation model, a) Flinders topcoat (left), backing coat (right); b) Brisbane topcoat and backing coat; c) Cairns topcoat and backing coat; d) Charleville topcoat and backing coat. Green line = remaining chromate, blue line = mass loss of zincalume, red line = mass loss of steel.



5.2.4 Conversion of Mass Loss to Life Estimate

The output from the holistic model is generally a mass loss per year for the metals and metallic coatings and the paint coatings provide a measure of the damage accumulation. In order to interface the holistic model with the CBR engine, the output needs to be converted into a component life, in years.

To convert the mass loss to a component life three additional pieces of information are required

1. Final Failure Criteria
2. Event "Tree" for failure
3. Conversion from mass loss per year to mass loss over an appreciable time.

5.2.4.1 Final failure Criteria

The failure criteria for a component depends on its use. Three types of criteria are relevant

1. Structural safety
2. Serviceability
3. Aesthetics

The criteria in each case may be different and needs to be applied separately for roof sheeting and guttering.

Roof sheeting

1) Structural safety – not relevant

2) Serviceability – no through sheet corrosion

3) Aesthetics:

   a) Light criteria – Red rust less than 50%

   b) Tight Criteria – No Red rust

Guttering

The criteria for failure would have the same definitions but the user may select different criteria for roof and guttering.

5.2.4.2 Event Tree For failure

The event tree for failure would be different for different materials and criteria. These required events are set out in Table 5.6 and definitions of these events in Table 5.7.

Table 5.6 Required Events for Failure

Material	Criteria	Event 1	Event 2	Event 3
Colorbond®	Serviceability	Failure of polymeric coating	Failure of Zincalume coating	Through Corrosion of Steel substrate
Colorbond®	Aesthetics-A	Failure of polymeric coating	Failure of Zincalume coating	50% Red Rust
	Aesthetics -B	Failure of polymeric coating	Failure of Zincalume coating	
Zincalume	Serviceability	Failure of Zincalume coating	Through Corrosion of Steel substrate	
	Aesthetics-A	Failure of Zincalume coating	50% Red Rust	
	Aesthetics -B	Failure of Zincalume coating		
Zincalume	Serviceability	Failure of Zinc coating	Through Corrosion of Steel substrate	
	Aesthetics-A	Failure of Zinc coating	50% Red Rust	
	Aesthetics -B	Failure of Zinc coating		

Table 5.7 Definition of Failure

Event	Definition	Explanation
Failure of polymeric coating	D= 1	D is damage index
Failure of Zincalume coating	ML= 0.75* Coating Mass	Coating mass is specified for all materials – assume Coating mass = 150 g/m2
Failure of Zinc coating	ML= 0.75* Coating Mass	Coating mass is specified for all materials – assume Coating mass = 275 g/m2
Through Corrosion of Steel substrate	TL= 1 * Component Thickness	Component Thickness is specified for all materials assume = 0.6 mm
50% Red Rust	TL=0.1 mm.	

5.2.4.3 Derivation of Events from Holistic Model

Polymeric Coatings

The holistic model gives D per year which is to be called d

$$D = d*T^{n} \quad \text{…Eqn 5.8}$$

Here T is time in years , n is a constant which can be taken as 1.1

Zincalume

The holistic model gives ML per year which is to be called m

$$ML = m*T^n \quad \text{…Eqn 5.9}$$

Here T is time in years , n is a constant which can be taken as 0.6.

Zinc Coating

The holistic model gives ML per year which is to be called m

$$ML = m*T^n \quad \text{…Eqn 5.10}$$

Here T is time in years , n is a constant that depends on m

$$n= 0.62+0.212ln(m/14.2),\ n>=0.62. \quad \text{…Eqn 5.11}$$

Steel

Holistic model gives ML per year which is to be called l

$$TL = C*l*T^n \quad \text{…Eqn 5.12}$$

Here T is time in years , C is a rust concentration factor (set at 2.5) n is a constant that depends on l

$$n= 0.62+0.212ln(l/14.2) \quad \text{…Eqn 5.13}$$

5.2.5 Gutter Survey

A roof and gutter survey carried out by CSIRO MIT has been used to determine some parameter values in the modified holistic model for gutters. The age and condition of a number of gutters were assessed and the data is reported in CRC report 2002-059-B No 10, Summary of Gutter Survey by CSIRO.

The buildings surveyed were located in 7-10 Km radius of CSIRO at Highett (Figure 5.2) which has a salt deposition of approximately 8 mg/m²·day and a corrosion rate for steel of approximately 10µm/year (See 5.2.1.1 above).

The buildings surveyed were of various construction types and the gutters were basically Colorbond®, galvanised, Zincalume or copper. The copper gutters were

ignored, they were in good condition after 25+ years and are not typical of current building practice. Gutters that did not have a painted coating on the inside, were deemed to be the corresponding base metal in type. That is some Colorbond® gutters were Colorbond® on the outside and so were deemed to be Zincalume. Also a number of the galvanised and Zincalume gutters were painted on the outside but not the inside, so they were typed as their metal type.

In general the all the gutters had some dirt present in them, most had a complete covering of the bottom or were full of dirt and leaf litter.

5.2.5.1 Damage Scale

In order to interpret the data a damage rating scale was conceived for the gutters. The scale went from 0 with no damage to 5 with perforation of the gutters. Table 5.8 details the rating scale used to rate the gutters.

Table 5.8 Legend of damage ratings for Gutter survey

Damage Rating	Condition	Condition around joints
0	No Damage	No Damage
1	Some loss of paint gloss/coating (Top coat only on multi-coat systems), dulling of surface	Discolouration of paint at joins and near rivets, fasteners or brackets
2	Loss of paint (chips lost, peeling, undercoat may still be intact), White corrosion product less than 50%	Some corrosion of rivets, fasteners or brackets
3	Some red rust present, less than 50% of a particular area ie, bottom surface	White corrosion products on rivets, fasteners or brackets and cut edges
4	50- 100% red rust	Red rust and white corrosion products on rivets, fasteners or brackets and cut edges
5	Perforation	Loss of rivets, fasteners or brackets, perforation of material

5.2.5.2 Summary of Gutter Data

The gutter information from the survey is summarised in Table 5.9. The corrosion level shown in the table is based on the Corrosion map of Melbourne 1979-80 (King, Martin and Moresby, 1982), and is in µm/year loss of metal from Low Alloy Copper Barring Steel coupons, a standard corrosion coupon used by CSIRO for corrosion mapping work. This gives an indication of the aggressiveness of the environment.

Table 5.9 Summary of gutter survey to April 2005

Name	Age (yrs)	Distance to Bay (km)	Longitude Latitude	Corrosion level (µm/year)	Gutter type	Painted		Gutter Condition			
						Outside	Inside	Bottom Inside	Inside sides	Outside	Around joins & cutouts
B1	<1	2.4	145° 0.84' E 37° 54.91' S	13	Colorbond®			0	0	0	0
D1	6.5	4	145° 7.06' E 37° 58.91' S	17	Zincalume®	Yes		3	2	0	0
C1	10	4.5	145° 3.52' E 37° 57.08' S	16	Zincalume®	Yes		3	1	0	0
M1	~15	1.8	145° 4.61' E 37° 58.75' S	17	Galvanized	Yes	Yes	3	1	0	
BR1	30	0.15	145° 0.85' E 37° 58.31' S	22	Galvanized	Yes		4	3	3	0
C2	25	2.5	145° 4.82' E 37° 58.34' S	17	Zincalume®	Yes		2	1	0	
P1	7	0.8	145° 4.67' E 37° 59.51' S	17	Colorbond®			0	0	0	4
B2	10-15	1.8	145° 0.42' E 37° 55.17' S	15	Zincalume®	Yes		2	2	0	0
H1	>30	4	145° 3.28' E 37° 57.16' S	15	Galvanized	Yes		4	4	Lip 4 rest 2	?
CY1	3	9	145° 7.81' E 37° 55.68' S	15	Zincalume®	Yes		0	0	0	0
B3	>20	0.8	144° 59.75'E 37° 53.75'S	15	Galvanized	Yes		4	2	2	
B4	30	0.9	144° 59.79'E 37° 53.61'S	15	Galvanized			5	3	5	2
B5	15	1.4	145° 0.06'E 37° 54.06'S	13	Zincalume®	Yes		3	2	0	2
B6	5	1.4	145° 0.06'E 37° 54.06'S	13	Colorbond®			1	0	0	3
A1	5	10	145° 5.56'E 37° 52.15'S	13	Colorbond®			0	0	0	2
CH214	26	3	145° 2.43'E 37° 57.02'S	17	Galvanized			4	3		
CH303	50?	3	145° 2.43'E 37° 57.02'S	17	Galvanized			4			
CH301	55	3	145° 2.43'E 37° 57.02'S	17	Galvanized			5	4	5	
CH201	48	3	145° 2.43'E 37° 57.02'S	17	Galvanized	Yes		5		5	
CH207	35	3	145° 2.43'E 37° 57.02'S	17	Copper						
CH208	27	3	145° 2.43'E 37° 57.02'S	17	Copper						
CH213	25	3	145° 2.43'E 37° 57.02'S	17	Zincalume®		Yes	4			
CH213a	16	3	145° 2.43'E 37° 57.02'S	17	Colorbond®			0	0	0	0
CH101a		3	145° 2.43'E 37° 57.02'S	17	Colorbond®			0	0	0	0
CH101	26	3	145° 2.43'E 37° 57.02'S	17	Copper box						
CH102	38	3	145° 2.43'E 37° 57.02'S	17	Galvanized	Yes		5		5	
CH106	<26	3	145° 2.43'E 37° 57.02'S	17	Colorbond®			0	0		3
CH109		3	145° 2.43'E 37° 57.02'S	17	Galvanized			5		5	
CH207a	<10	3	145° 2.43'E 37° 57.02'S	17	Colorbond®			0	0	0	0
CH208a	<10	3	145° 2.43'E 37° 57.02'S	17	Colorbond®			0	0	0	0
CH41a	<10	3	145° 2.43'E 37° 57.02'S	17	Colorbond®			3	3	3	

Name	Age (yrs)	Distance to Bay (km)	Longitude Latitude	Corrosion level (µm/year)	Gutter type	Painted		Gutter Condition			
						Outside	Inside	Bottom Inside	Inside sides	Outside	Around joins & cutouts
CH25	28	3	145° 2.43'E 37° 57.02'S	17	Galvanized			5		5	
CH11	46	3	145° 2.43'E 37° 57.02'S	17	Galvanized	Yes		5	4	5	
CH27	35	3	145° 2.43'E 37° 57.02'S	17	Galvanized	Yes		5		5	
CH29	10-15	3	145° 2.43'E 37° 57.02'S	17	Colorbond®			0	0	0	2
CH29a	10-15	3	145° 2.43'E 37° 57.02'S	17	Colorbond®			0	0	0	2
CH13	38	3	145° 2.43'E 37° 57.02'S	17	Galvanized	Yes		5		5	
CH13a	10	3	145° 2.43'E 37° 57.02'S	17	Colorbond®			0	0	0	0
CH13b	10	3	145° 2.43'E 37° 57.02'S	17	Colorbond®			3	1	1	
CH12a	10	3	145° 2.43'E 37° 57.02'S	17	Colorbond®			0		1	3
CH14	38	3	145° 2.43'E 37° 57.02'S	17	Galvanized	Yes		5		5	
CH15	20	3	145° 2.43'E 37° 57.02'S	17	Colorbond®			0		2	
CH15a	37	3	145° 2.43'E 37° 57.02'S	17	Galvanized	Yes		3			
CH15a	10?	3	145° 2.43'E 37° 57.02'S	17	Colorbond®			3	0	0	
CH35	27	3	145° 2.43'E 37° 57.02'S	17	Zincalume®	Yes	No	2	2	2	2

The corrosion levels are all between 13 and 22µm/year. These are typical values for Melbourne and not considered severe.

The data collated in Table 5.9 is shown graphically in Figure 5.15, Figure 5.16 and Figure 5.17.

Figure 5.15 Graphical representation of the state of Galvanised gutters with age

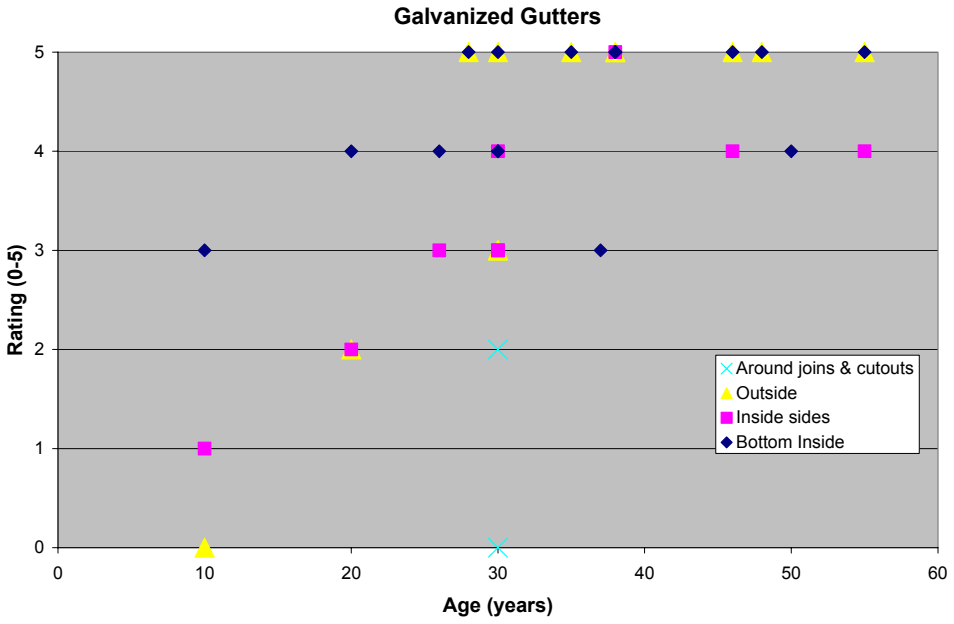


Figure 5.16 Graphical representation of the state of Zincalume gutters with age

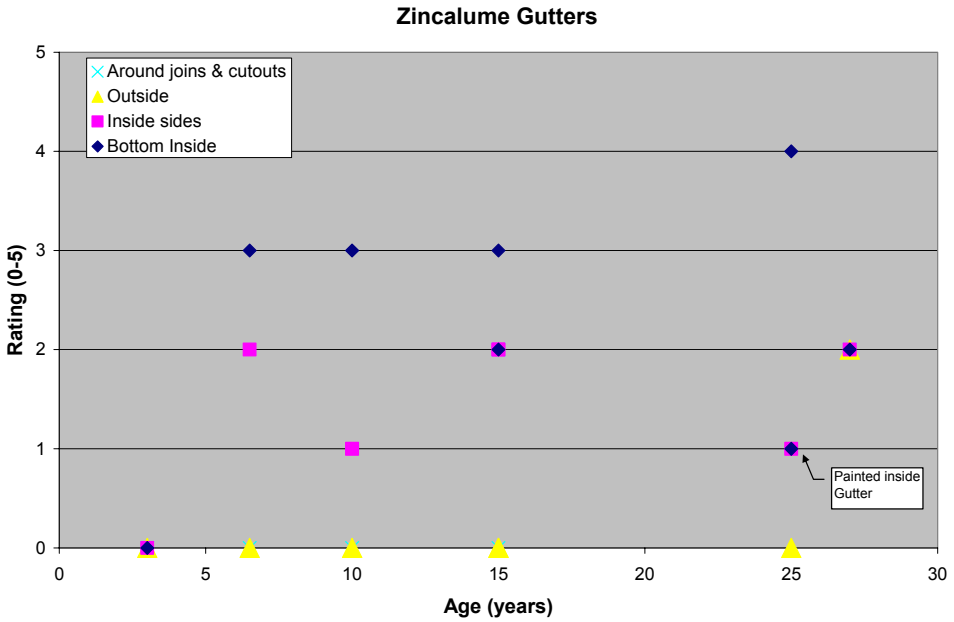
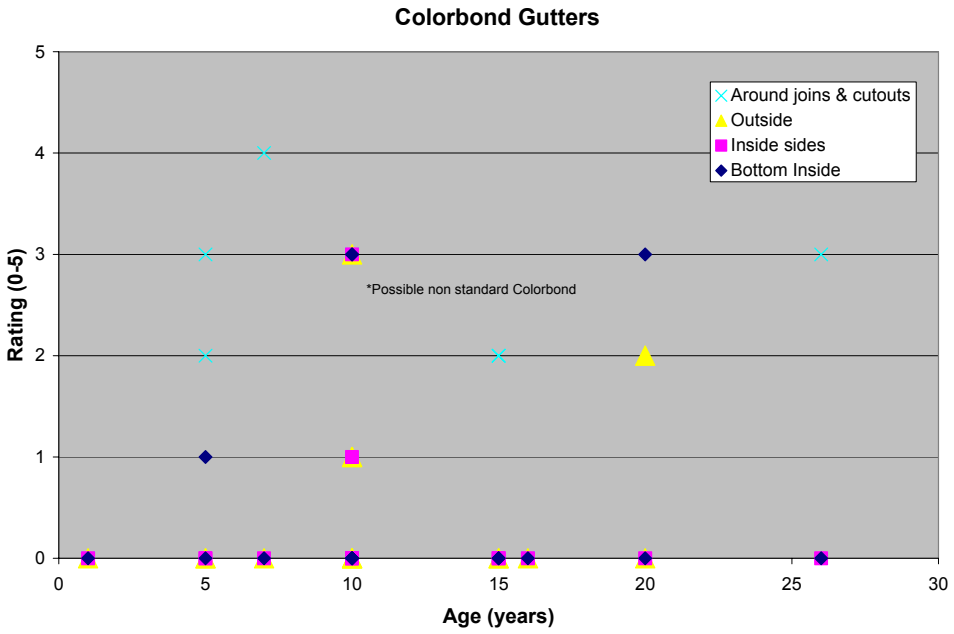


Figure 5.17 Graphical representation of the state of Colorbond® gutters with age



The graph for the galvanised gutters shows ages up to 60 years while the Zincalume® and Colorbond® graphs only show up to 30 years. While Colorbond® was introduced into Australia in the mid 1960s, it was only coated on one side and used galvanised steel as a base metal. Zincalume® was introduced to Australia in the 1970s and Colorbond® at that time changed to having Zincalume® as the base material and was coated on both sides. This is why we only have data up to 30 years for Colorbond® and Zincalume®. During this time numerous Colorbond® paint systems have been used with each one having slightly different weathering properties.

Looking at the galvanised gutter graph, Figure 5.15, the gutters show some damage between 10 and 20 years and significant damage requiring replacement after approximately 20 to 30 years. It should be noted that the youngest galvanised gutter surveyed was 10 years old.

The Zincalume® gutters, Figure 5.16, show some damage after 7 to 10 years with one gutter showing significant damage requiring replacement at 25 years.

The graph of the Colorbond® gutters, Figure 5.17, shows damage on some of the gutters after 5 to 10 years with some showing spots of red rust. The oldest gutters were 20 to 25 years old and none of the gutters showed enough damage to consider replacement.

5.2.5.3 Conclusions from Survey

From this gutter survey it is concluded that galvanised gutters in the survey area show some damage by 10 years, with significant damage, leading to the need for replacement, at around 20 years. The Zincalume® gutters showed some damage around 10 years and replacement was needed after 25 years. The Colorbond® gutters showed some damage after 5 years, but no significant damage causing the need for replacement in any of the gutters surveyed with the oldest being approximately 20-25 years.

5.2.6 Holistic Model Program for Gutters

The modifications made to the holistic model to adapt it for use with gutters have been incorporated into a stand-alone program, mainly for development purposes, but it can be used to model mass loss for gutters at any point in Australia.

The Holistic model as outlined previously contains a number of modules (Figure 3.5) to:

a) predict the salinity at a location

b) predict the climate at a location

c) predict salinity retention on a component on a building

d) predict the state of a surface on a component on a building

e) predict the damage of the component on the building.

In adapting the holistic model for the gutter application:

a) and b) were unchanged from the prior holistic model,

c) modifications were made to constants in the model to reflect the different cases for gutters but the basic formulation remained the same,

d) modifications were made for the case of a gutter filled with dirt and debris (TOW),

e) modifications made for galvanised steel and zincalume and completely developed for Colorbond.

5.2.6.1 Salinity Retention

In calculating whether salt will be retained on a surface in the event of rain it is assumed that salt cleans off a surface according to the following relationships:

$$D_i \text{ after wash} = \Phi + \psi^* D_{i-1} \quad \text{...Eqn 5.14}$$

Where Di is the retained salt after a rain event and Di-1 is the deposited salt prior to a rain event . Φ is taken as 1 and the values of ψ are given in Table 5.10. Here LMI, SMI and HMI refer to low ,medium and high moisture index which is a parameter which describes the rate of evaporation and O refers to open exposure (gutter bottom and edges) and S to sheltered (underside of gutter).

Table 5.10 Values of ψ defined for various parameter combinations

Moisture Index	Open/Sheltered	ψ
LMI	O	0.1
	S	0.6
SMI	O	0.5
	S	0.6
HMI	O	0.5
	S	0.6

5.2.6.2 State of surface of building component

Three states of a surface are defined

a) S1 – dry
b) S2 –wet from wetting of hygroscopic salts
c) S3 - wet from rain

The holistic model calculates state on a three hour interval. The standard model assumes that a surface is in state 3 whenever rain is occurring but once the rain has ceased, it is dry before the next 3 hour period .If the rain ceased in the middle of the last time period this implies drying takes no more than 1.5 hours. The studies of gutters indicates that this is a reasonable assumption for all cases, except the bottom of gutters filled with dirt and debris. For this case it is assumed that the gutter remains in State 3 for 48 hours after rain.

5.2.6.3 Damage to Components

The damage to components is also calculated each three hours from a knowledge of the state of the component, the retained salinity and climatic parameters. Two different approaches are used for a) uncoated metals (steel, galvanised steel and zincalume) and b) coated steel.

Uncoated Metals

The standard holistic methods is used in which the corrosion rate is calculated each three hours according to the following equations:

$$Ms_1 = 0 \qquad \text{…Eqn 5.15}$$

$$Ms_2 = \zeta * M_2 \qquad \text{…Eqn 5.16}$$

Where M_2 depends on RH

For 35<RH<75

$$M_2 = 3 + \Phi * D^{\Phi} \qquad \text{…Eqn 5.17}$$

Where D is the retained salt and the values of the constants are given in the Table 5.11– Table 5.16.

For RH>75

$$M_2 = \Theta + \Omega * D^{\Psi} \qquad \text{…Eqn 5.18}$$

For State 3

$$Ms_3 = \zeta * M_3 \qquad \text{…Eqn 5.19}$$

In the case of M_3, the rate of mass loss varies on the basis of the component case (gutter – edge, bottom(cleaned and uncleaned) or sheltered). This approach is based on the understanding that significant salt will be retained in the dirt at the bottom of the gutter and this will significantly increase the corrosion rate for gutter bottoms in an uncleaned condition. In fact in a future version of the model it would be desirable to introduce a retained salt dependence into M_3 and then remove this component case dependence.

Table 5.11 Constants for galvanised steel mass loss in State 2

Θ	0.02
Ω	0.027
Ψ	0.5
Ʒ	0.02
θ	0.027
Φ	0.5
ζ	1

Table 5.12 Constants for galvanised steel mass loss in State 3

	ζ
open	1
sheltered	2
Partial sheltered	1.5

Table 5.13 Additional constants for galvanised steel mass loss in State 3

Case	**Ms3**
Gutter-sheltered	0.02
Gutter-open -edge	0.05
Gutter-open bottom-uncleaned	0.6
Gutter –open bottom-cleaned	0.05

Table 5.14 Constants for Zincalume mass loss in State 2.

Θ	0.027
Ω	0.004
Ψ	0.5
Ʒ	0.0
θ	0.002
Φ	0.5
ζ	1

Table 5.15 Constants for Zincalume mass loss in State 3

	ζ
open	1
sheltered	2
Partial sheltered	1.5

Table 5.16 Additional constants for Zincalume mass loss in State 3.

	Ms3
Gutter-sheltered	0.027
Gutter-open -edge	0.05
Gutter-open bottom	0.15
Gutter –open bottom-cleaned	0.05

5.2.6.4 Application of the Model

To test the model, the mass losses at two locations in southern Queensland were estimated. One was a Marine location and the other a benign location (Table 5.17). In Table 5.18 a comparison of the estimate of the life of gutters based on the Delphi study, roof survey and holistic model is made. It is apparent that the lifespan estimates are similar when like cases are considered.

Table 5.17 Estimated mass loss at two locations in Queensland

Longitude	Latitude	Salinity	Exposure				Mass loss – g/m^2.
153 441	28061	38	Open	bottom	zincalume	NC	19
			Open	bottom	zincalume	C	13.5
			sheltered		zincalume		7.3
			Open	edges	zincalume		8.9
153 441	28061	38	Open	bottom	galvanised	NC	33
			Open	bottom	galvanised	C	31
			sheltered	zincalume	galvanised		31
			Open	edges	galvanised		18
153 425	28049	6	Open	bottom	zincalume	NC	67
			Open	bottom	zincalume	C	16
			sheltered		zincalume		9
			Open	edges	zincalume		12
153 425	28049	6	Open	bottom	galvanised	NC	56
			Open	bottom	galvanised	C	18
			sheltered	zincalume	galvanised		11
			Open	edges	galvanised		18

Table 5.18 Comparison of Gutter Life by Model and other Methods

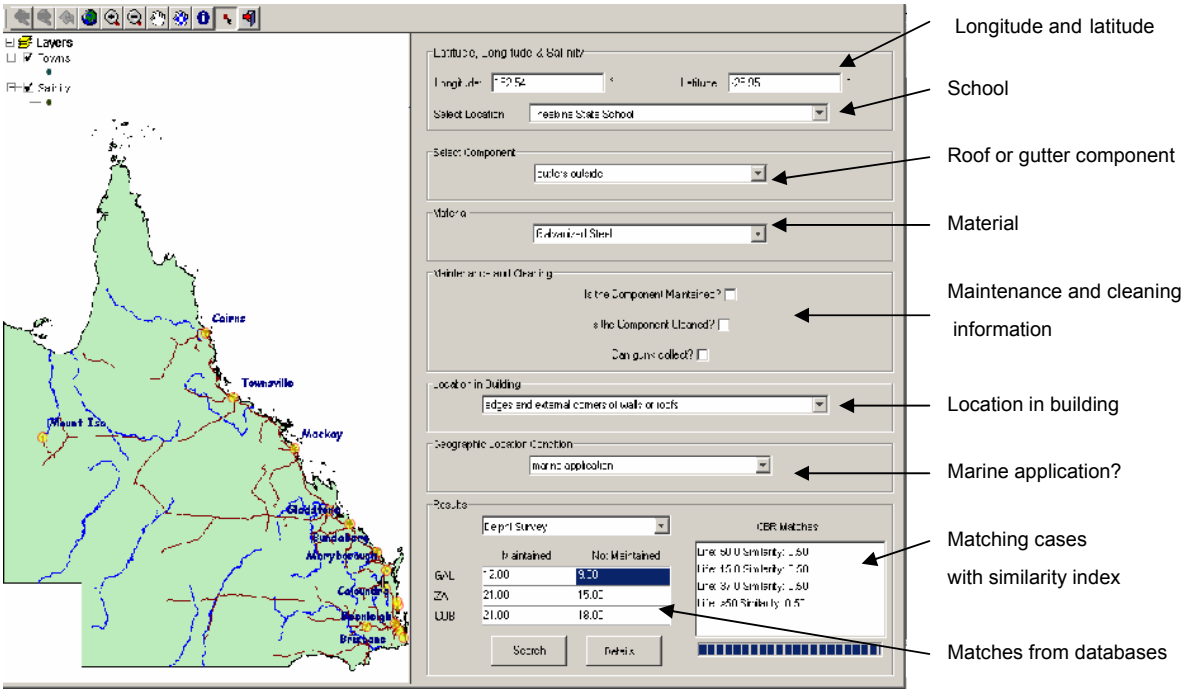
Location	Component Case	Method	Life
Marine	Unspecified position, galvanised	Delphi	10
Benign	Unspecified position galvanised	Delphi	32
Marine	Unspecified position Zincalume	Delphi	21
Benign	Unspecified position Zincalume	Delphi	42
Marine	Unspecified position galvanised	Survey	15
Benign	Unspecified position galvanised	Survey	55
Benign	Unspecified position zincalume	Survey	>40
Marine	Sheltered-galvanised	Holistic Model	15
Marine	Internal Edge-galvanised	Holistic Model	33
Marine	Internal –bottom –cleaned-galvanised	Holistic Model	15
Marine	Internal –bottom –not cleaned-galvanised	Holistic Model	14
Benign	Sheltered-galvanised	Holistic Model	33
Benign	Internal Edge-galvanised	Holistic Model	>60
Benign	Internal –bottom –cleaned-galvanised	Holistic Model	33
Benign	Internal –bottom –not cleaned-galvanised	Holistic Model	7
Marine	Sheltered-zincalume	Holistic Model	24
Marine	Internal Edge-zincalume	Holistic Model	37
Marine	Internal –bottom –cleaned-zincalume	Holistic Model	16
Marine	Internal –bottom –not cleaned-zincalume	Holistic Model	5
Benign	Sheltered-zincalume	Holistic Model	37
Benign	Internal Edge-zincalume	Holistic Model	50
Benign	Internal –bottom –cleaned-zincalume	Holistic Model	21
Benign	Internal –bottom –not cleaned-zincalume	Holistic Model	13

5.3 CBR Queensland Schools' Gutter User Interface

A GUI has been created to allow users to interrogate the CBR program developed for the gutters in the Queensland Schools application (Figure 5.18). A subset of schools in the Southern coastal regions has been used in the program and can be accessed through a drop down menu. A red cross will indicate the position on the map of Queensland. If a school is not chosen, the map of Queensland can be used to select points within the state which will define the longitude and latitude.

Dropdown menus have been incorporate to allow selection of gutter components and materials etc. Check boxes define whether the component under consideration is Maintained, or Cleaned etc. The search button at the bottom initiates the CBR engine and matching cases are retrieved and shown in the bottom right window, with the corresponding similarity index. Database matches are also shown in the table to the left; all three gutter material types are listed. A button at the bottom of the window can be used to get further details of the matching cases listed.

Figure 5.18 GUI developed for the Queensland schools' gutter application



5.4 Utility of Present Results

As demonstrated in Section 5.2.6 with the comparison of results from different sources, there is good correlation between the various methods. However, before the modified holistic model is released commercially, it requires more verification and collection of data on maintenance and lifespans of gutters of the different materials.

6. QDMR APPLICATION

Maintenance of bridge structures is a major issue for the Queensland Department of Main Roads so this was chosen as the focus of the application for this industry partner. The general cases defined for the CBR engine are based around building components. These are obviously not directly applicable to metallic components in bridges. Therefore, any CBR driven program for corrosion in bridges will require the definition of bridge elements (relevant to bridges constructed in Queensland) to be used as the basis for case construction and comparison.

Five representative bridge structures were provided by the QDMR and analysed. The five bridges are the Gladstone Port Access Road Overpass, Stewart Road Overpass, South Johnstone River Bridge, Johnson Creek Bridge and the Ward River Bridge. Common elements were determined so that the results can be interpolated across the range of bridge types in Queensland. This work is reported in detail in CRC Report 2002-059 No 9, Salt Deposition on Queensland Bridges, prepared by David Paterson.

6.1 Analysis Methodology

The salt deposition on the five representative bridge structures was computed using computational fluid dynamics (CFD). Upstream and ground boundary conditions were derived from the Geographical Information System (GIS) for salt deposition and metal corrosion in Australia. The results have been summarised by dividing by the deposition that would occur on a salt candle at the same location and by averaging over a set of physical locations (zones) on each bridge.

6.1.1 Computation Method

CFD is basically the solution of the conservation of mass and momentum equations on computer. The conservation of momentum in fluids is known as the Navier-Stokes equation. This is a partial differential equation with three vector components (x, y, z). The equations are solved in averaged form because the time-dependent details of turbulence can't be resolved. This adds the term for the mean square turbulence into the momentum equations, and a turbulence model is used to solve for the mean square turbulence.

In mathematics, the mean square turbulence is $2k/3$, the turbulence intensity is $\sqrt{2k/3}$ divided by the mean velocity, and the rate of dissipation of k is called ε. The k-ε turbulence model is a very old model that has become the industry standard. It has two extra partial differential equations, one each for k and ε. The Re-Normalisation Group (RNG) turbulence model is a more modern variant based on the mathematics of re-normalisation familiar from quantum mechanics.

The solution of these partial differential equations was done using the commercial CFD program CFX 5.7. CFX 5.7 uses a finite volume analysis method with an unstructured grid. The finite volume method is similar to the finite-element method commonly used in structural engineering, but differs in enforcing exact conservation of convected quantities on pre-

defined volumes/elements. Turbulence was computed using the k-ε model and this was compared with the RNG turbulence model for two bridges.

For each bridge approximately 100,000 salt spray aerosol particles were released upstream of the bridge section and tracked using Lagrangian methods. Deposition rates in mg m^{-2} were calculated automatically from this by averaging over small elements of the bridge surface. Results were checked by trying a range of different aerosol release areas and release strategies (uniform vs random) for each bridge.

The aerosol particles were assumed to be statistically random in diameter, with prescribed mean, standard deviation and the volume of the droplets was described by a normal distribution. This approach correctly simulates the total salt deposition without the need to model exorbitant numbers of very small particles.

It was not assumed that the air flow over each bridge was free from vortex shedding, but that proved to be the case for each of the final simulations reported here.

The assumed roughness of the concrete for each bridge is 0.3 mm. This only has a small influence on the deposition of salt on the bridge near the coast or when the relative humidity is high (as assumed here) because the moisture in the transported aerosol sticks the salt to the concrete. The assumed roughness has a large influence when the salt deposition rate is very low at low relative humidities (below 33%).

6.1.2 Information Supplied by QDMR and extracted from the GIS

The DMR supplied drawings and aerial photographs on each of the five bridges. They also supplied locations, latitude and longitude and environmental data for each bridge. The GIS provided the distance to the coast and the representative local vegetation height (allowing for urban development).

Mean wind data is extracted from climatic information from the Australian Bureau of Meteorology. The data used is the morning and afternoon wind speeds and directions for the four seasons at adjacent weather stations. Sea breezes are accounted for by the difference between morning and afternoon data.

The distance to the coast was also calculated from maps and atlases and compared with two different sets GIS data. For bridges within four kilometres of the coast there was often a significant discrepancy between the three figures, and the one thought to be most accurate was selected.

The GIS database for vegetation contains the primary vegetation type, the fractional ground cover by this vegetation, and the understorey vegetation type. It also contains urban development details if they're relevant. From this, algorithms developed at CSIRO were used, together with the mean wind velocity, to determine the wind velocity profile and turbulence intensity at each site. The vegetation was assumed to be of uniform height for the purposes of computation.

The bridge height above mean water level was used in conjunction with the wind velocity profile in setting the upstream wind speed and maximum turbulence length scale.

The mean and standard deviation salt aerosol particle diameter were calculated from an algorithm developed at CSIRO based on the distance from the coast and the effect of the size of Australian weather systems on the length of the wind path.

A constant relative humidity of 70% was used to calculate the density of the salt-containing aerosols. This is a typical relative humidity experienced in Queensland.

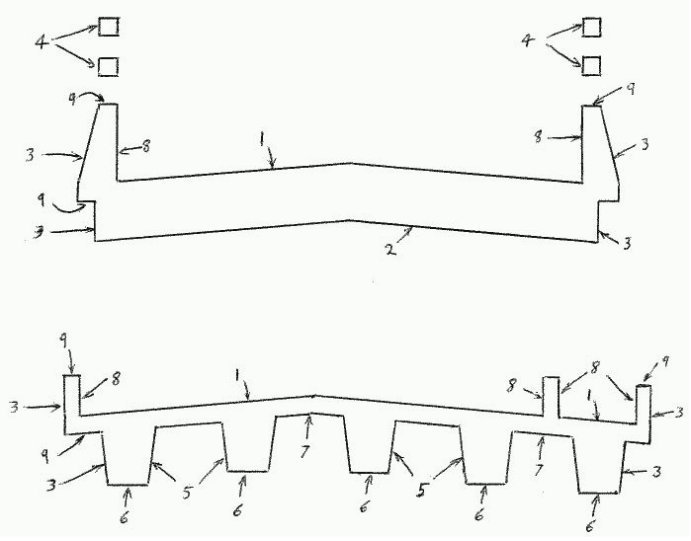
6.1.3 Defining Common Elements

The deposition of salt on any structure depends on two independent processes. The first is the transport of salt aerosol to the vicinity of the structure and the second is the effect of the shape of the structure on the deposition rate. The first of these can be measured by a salt candle. The salt deposition measured by a salt candle at any location can be reliably extracted from the GIS model of metallic corrosion.

For each bridge, a separate computation was done of deposition on a salt candle at the same location. The ratio of the deposition on the bridge to that on the salt candle quantifies the effect of the shape of the structure on the deposition rate.

For the comparison of different bridge superstructures, results were averaged over a set of physical locations (zones) on each bridge. These zones are shown for two typical bridge cross sections in Figure 6.1.

Figure 6.1 The layout of zones on two typical bridge cross sections.



1. Road surface and median strip
2. Bridge undersurface
3. Side face
4. Handrails
5. Side of support beams
6. Undersurface of support beams
7. Protected undersurface
8. Lane divider and inside the parapet
9. On top of the parapet and under the side overhang

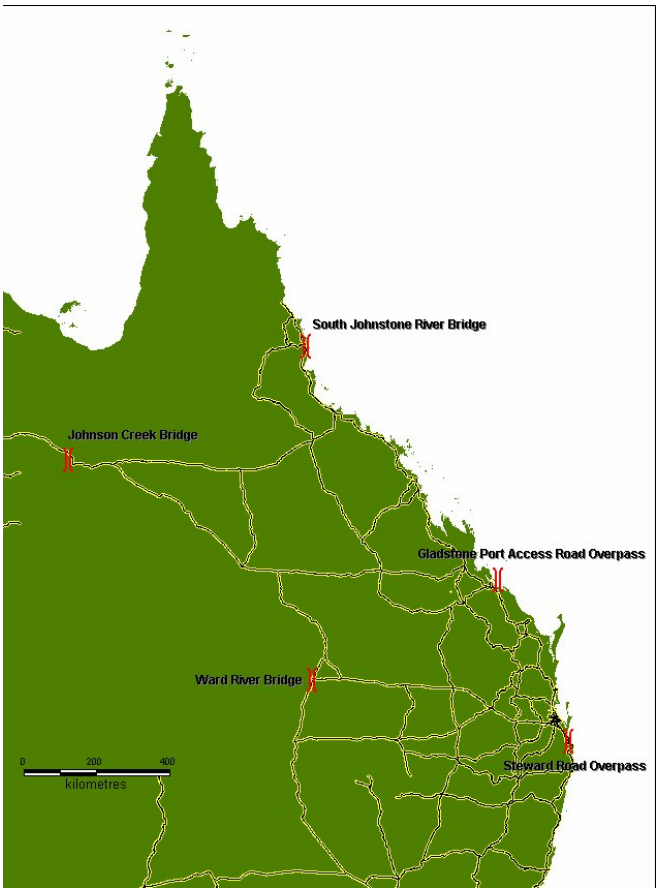
If the support beams are closer than 100 mm apart then "2" applies instead of "5, 6 and 7".

The deposition rates in these zones can depend on the detailed bridge design. Some zones will have similar deposition rates.

6.2 Analysis of the Five Bridges

The five bridges are the Gladstone Port Access Road Overpass, Stewart Road Overpass, South Johnstone River Bridge, Johnson Creek Bridge, and Ward River Bridge (see Figure 6.2).

Figure 6.2 Locations of the five bridges analysed



For the bridges described below, the abbreviation DSC means “times the deposition on a salt candle away from obstacles at the same location”. Table 6.1 gives a summary of the computed results. For approximate Zone locations see Figure 6.1; for details see each bridge in turn.

The salt deposition is influenced by the height to width ratio (H:W) of the superstructure. There is a critical H:W ratio, similar to that of the bridge over the Ward River, that maximises the salt deposition on the downwind side of the superstructure (Zone 3 landwards). The H:W ratio for the Johnson Creek Bridge is intermediate between that of the Stewart Road Overpass and the South Johnstone River Bridge and all use deck units; that explains why the deposition on Zone 2 for the Johnson Creek Bridge is intermediate between the other two.

Table 6.1 A summary of computed results; salt depositions on the 9 zones for the 5 bridges in DSC. u′/U is the upstream turbulence intensity and H:W is the height to width ratio of the superstructure.

		Gladstone	Stewart	Sth Johnstone	Johnson	Ward
	u' / U	0.29	0.18	0.12	0.41	0.3
	H:W	1:3.7	1:13.3	1:5.7	1:7.3	1:4.1
	1	0.65	0.19	0.79	0.36	1.13
	2		0.27	0.97	0.58	0.89
	3 seawards	1.30	1.47	1.46	1.66	1.22
	3 landwards	0.27	0.11	0.69	0.08	0.72
	3 average	0.79	0.79	1.08	0.87	0.97
Zone	4			2.53	1.59	
	5	0.38				0.37
	6	1.03				1.06
	7	0.18				0.87
	8	0.44	0.41	0.55	0.95	0.66
	9	0.50	0.69	0.80	0.80	0.78

The upstream turbulence intensity (u' / U) influences salt deposition in conjunction with the bridge roughness, the mean aerosol size and the relative humidity. For a smooth surface (eg. glass) with small aerosols (< 3 μm in diameter) the salt deposition rate can be so small as to be negligible. The same can be true when the relative humidity is low (< 33%). In these computations the surface is assumed to be rough enough and the relative humidity high enough for salt deposition to occur. In this case the deposition rate depends critically on u' / U, particularly when the aerosols are small. However, the effect of u' / U affects both the bridge and salt candle so the DSC value is relatively unchanged.

The salt deposition is also influenced by the structural details. For instance, the girders are further apart at Ward River than at Gladstone and this largely explains the difference in deposition between the girders (Zone 7). The high parapets on the Stewart Road Overpass help to explain the low deposition rate on the road surface there (Zone 1).

6.2.1 Gladstone Port Access Road Overpass

The Gladstone Port Access Road Overpass in Gladstone City is located at latitude 23°51' and longitude 151°30'. It is on the Gladstone Port Access Road between Glenlyon Road and the Port Precinct and passes over the top of Auckland Street and the railway lines. There is ocean to the North, North East and East of this bridge.

The bridge comprises twelve spans ranging in length from 28.4 metres to 37 metres. The superstructure consists of a reinforced concrete deck on rectangular prestressed concrete deck units for span 12 and on five T-ROFF trough-shaped prestressed concrete girders for spans 1 to 11. For these 11 spans the total width of the superstructure is 10.44 metres and the height is 2.81 metres, giving a height to width ratio of 1:3.7.

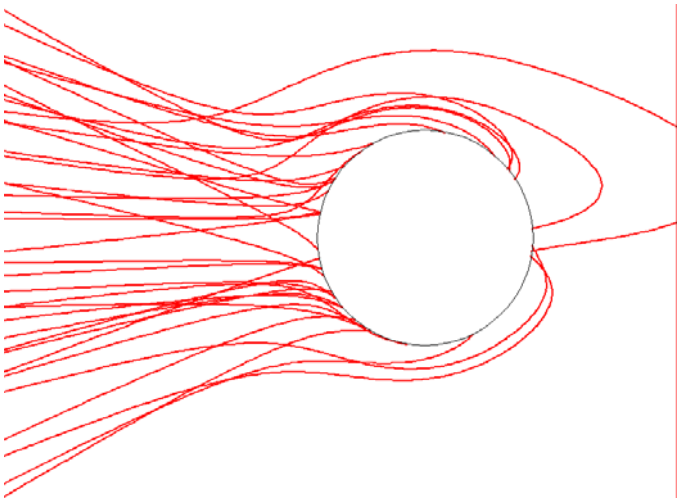
The salt deposition on the superstructure deck section for spans 3 to 5 and 9 to 11 was modelled on computer. This has a shorter parapet than spans 1, 2, 6, 7 and 8. In the second set of spans the salt deposition on the roadway is expected to be less because the road deck is protected by the higher parapet whereas the deposition on the underside is expected to be much the same. A preliminary computer simulation for spans 1, 2, 6, 7 and 8 suggested that it may be associated with vortex shedding in some winds.

The bridge height of the bridge deck in the centre spans varies from about 8.3 to 9.6 metres above ground level. In the simulations it is assumed to be 9.3 metres above ground level.

The salt deposition on a salt candle, extracted from our GIS database at the location of Gladstone for a marine environment at the latitude and longitude given, is 13.3 $mg.m^{-2}.day^{-1}$. This does not take into account the bridge height.

The deposition on the salt candle for each bridge was also computed. Tracks of 35 of the 2678 particles that deposited on the salt candle for Gladstone are shown in Figure 6.3. In all, 50,000 aerosol particles released upwind. The salt candle has a diameter of 25.4 mm. The flow domain is 300 mm long by 200 mm high. The grid contains 86,000 elements. It extends out of this plane a distance of 13 mm.

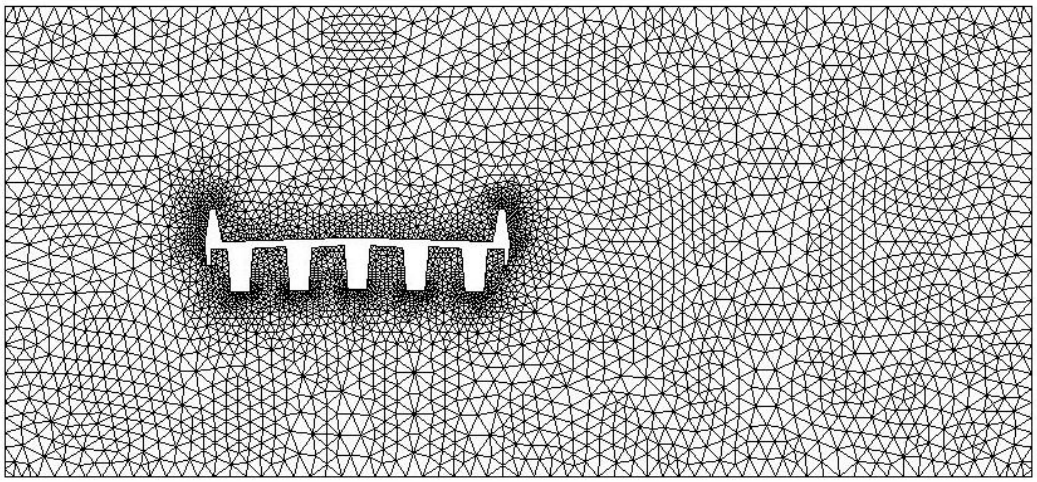
Figure 6.3 Tracks of 35 particles deposited on a salt candle in the same flow conditions as those of the Gladstone Port Access Road Overpass. Wind flow is from left to right



More salt is deposited on the front of the candle than the back. The cylindrical candle shelters the region directly downstream, leading to a low salt concentration in the air there. A pressure wave upstream of the cylinder increases the salt concentration there above ambient levels.

Figure 6.4 shows the flow domain size and grid resolution used for simulation of wind flow around the superstructure of the Gladstone Port Access Road Overpass. The flow domain is 35 metres long by 16 metres high. The flow is from left to right. The domain is extended in the downwind direction to stop the downwind boundary conditions on velocity and turbulence feeding back into the flow and altering the pressure distribution around the structure; this is standard practice. The grid is finer around the bridge. The grid contains 86,000 elements. It extends out of this plane a distance of 0.5 metres.

Figure 6.4 The flow domain size and grid resolution used for the superstructure of the Gladstone Port Access Road Overpass



The deposition on the superstructure was checked using three different aerosol release strategies. In one aerosols were released directly upwind of the bridge, in the second they were released in bands above and below the bridge, in the third they were released over a broad area. Results for Gladstone are shown in Figure 6.5. The aerosol was diffused upstream due to turbulence.

Figure 6.5a shows the volume fraction of salt of aerosols that were released just upstream of the bridge, within 1.4 metres of the mid-height. The salt is transported first to the leading side of the parapet and much salt is deposited there. The flow separates from the top of the parapet, leaving an open recirculation region (shown in blue) behind it over the bridge deck. The flow separating from the bottom edge of the parapet hits the lower portion of the first T-ROFF girder. The flow separates again from the bottom of the first girder.

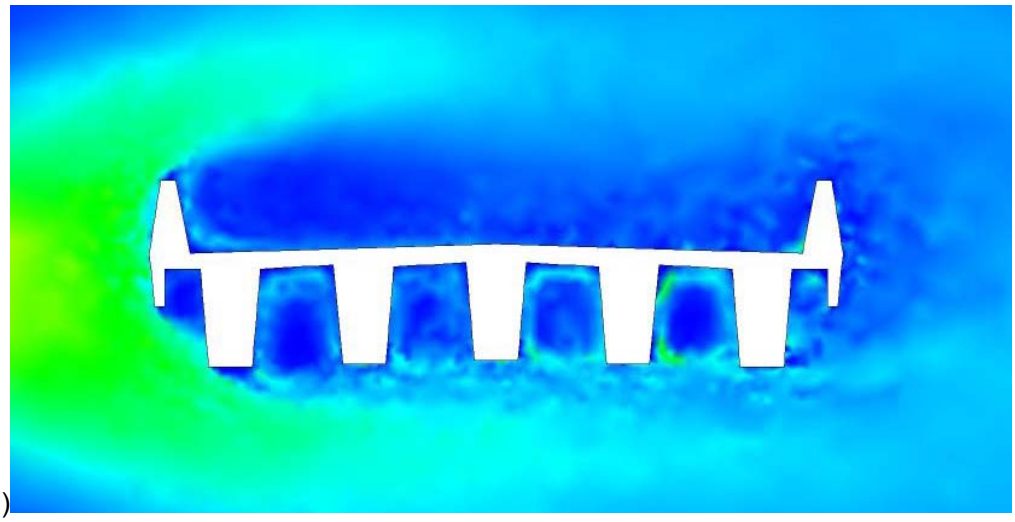
In Figure 6.5a both the turbulence in the mean wind and that generated by the bridge itself slowly brings salt into the recirculation regions. Over the bridge deck, it is seen that this increases the deposition on the leeward side of the bridge deck. Salt becomes trapped in the recirculation regions between the bridge girders, but although the concentration of the salt in the air between the girders is high, not much of it is deposited on the girders and the underside of the deck. The turbulence also brings some salt back onto the back of the downwind parapet, but not much.

Figure 6.5b shows the volume fraction of salt when aerosols were released between 1.4 and 2.8 metres of mid-height. There is less deposition on the upwind side of the parapet because the high pressure there deflects the salt away. More salt is trapped in the recirculation region over the front of the bridge deck. More salt is trapped between the second and third bridge girder.

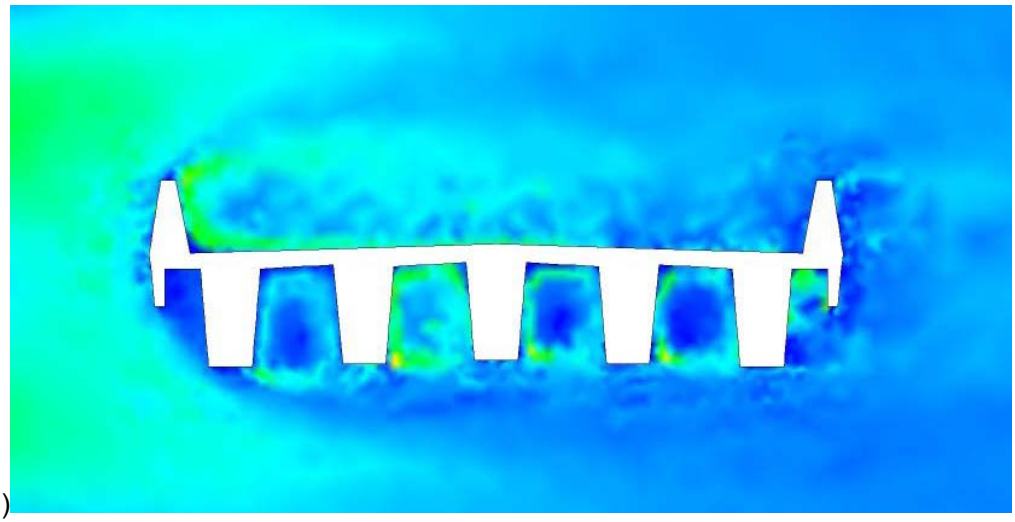
Figure 6.5c shows the sum of the top two figures plus salt released further from mid-height. The locations of the recirculation regions are now less clear. Salt concentrations in the air are now seen to be highest behind three of the girders and the leading parapet but the actual depositions in these regions, as shown by the colours on the surfaces, is still quite low.

Figure 6.5 **Volume fraction of salt around the superstructure of the Gladstone Port Access Road Overpass; a) particles released within 1.4 metres of the mid-height, b) particles were released between 1.4 and 2.8 metres of mid-height, c) all salt aerosol particles. Flow is from left to right. Red is high concentration and blue is low concentration.**

a)



b)



c)

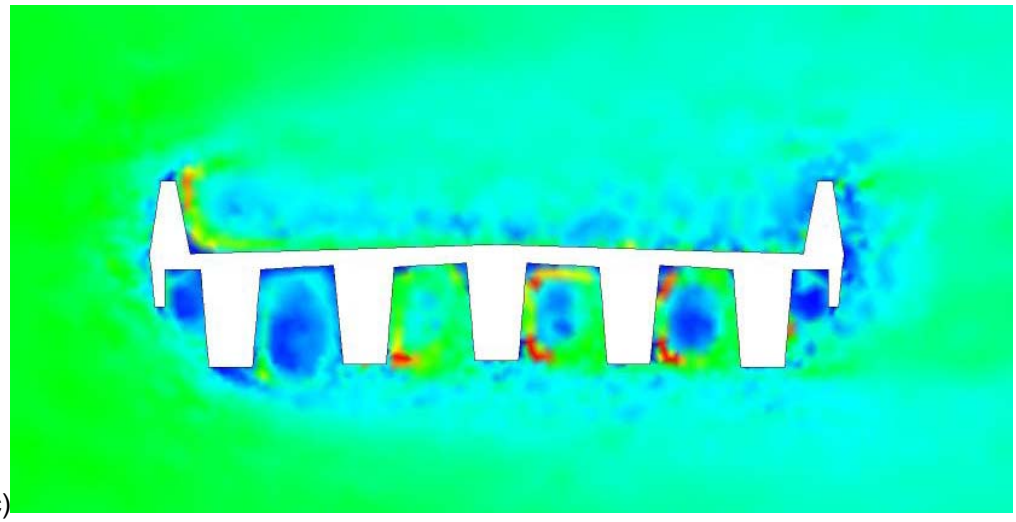


Figure 6.6 shows the locations of the zones used in analysing the deposition on the superstructure of the Gladstone Port Access Road Overpass. This is referred to in the subsequent graphs of salt deposition and in Table 6.1.

Figure 6.6 The locations of the zones for the superstructure of the Gladstone Port Access Road Overpass

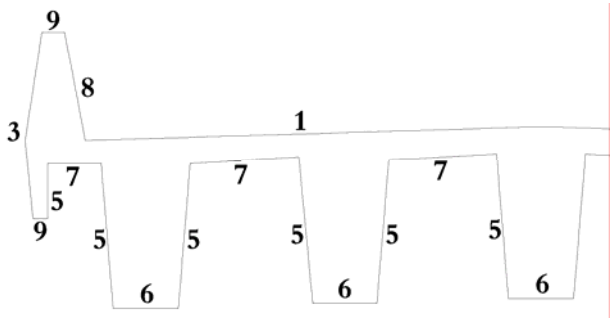


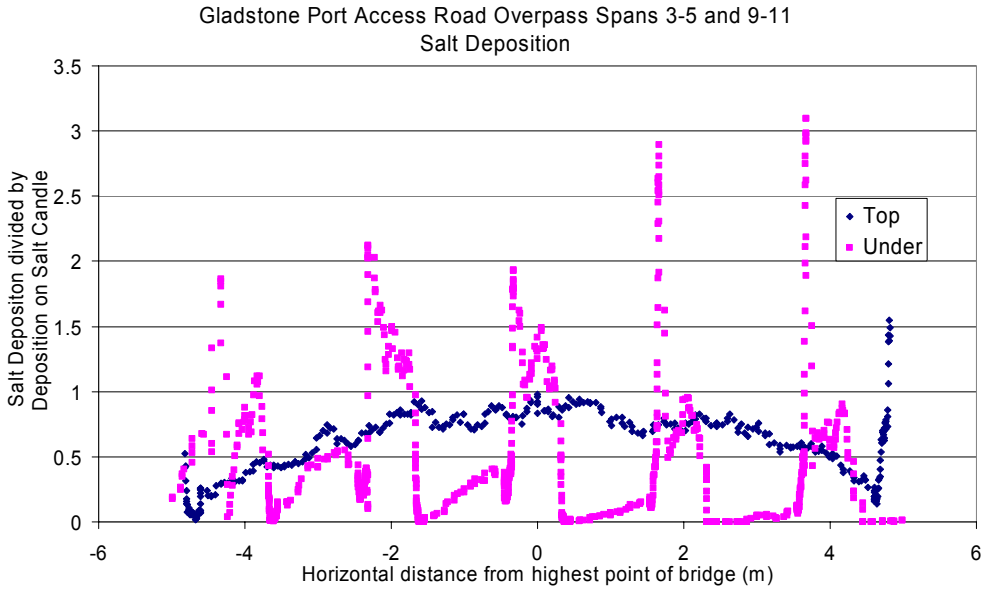
Figure 6.7 shows the salt deposition on the Gladstone Port Access Road Overpass, measured relative to the salt candle deposition of 13.3 mg m^{-2} day^{-1} at the same location. In Figure 6.7a the line marked “top” includes Zones 1 and 8, i.e. the top of the bridge deck and the inside of the parapets. The deposition is highest on the inside face of the downwind parapet. The deposition on the top of the bridge deck reaches a maximum of about 0.95 DSC in the middle of the deck, where DSC means “times the deposition on a salt candle away from obstacles at the same location”.

The line marked “under” includes Zones 5, 6 and 7, i.e. the underside of the bridge deck and all sides of the trough-shaped girders. The spikes on the graph represent the greatest salt deposition on the upwind faces of the girders. These are largest near the bottom of the girders. The bottoms of the girders have a salt deposition comparable in magnitude to that on the top of the deck, e.g. that below the centre of the bridge has an average deposition of 1.2 DSC. The deposition in the protected areas of the underside of the bridge deck between the girders varies from near zero in the downwind half of the superstructure to 0.55 DSC three metres upwind of the bridge centre.

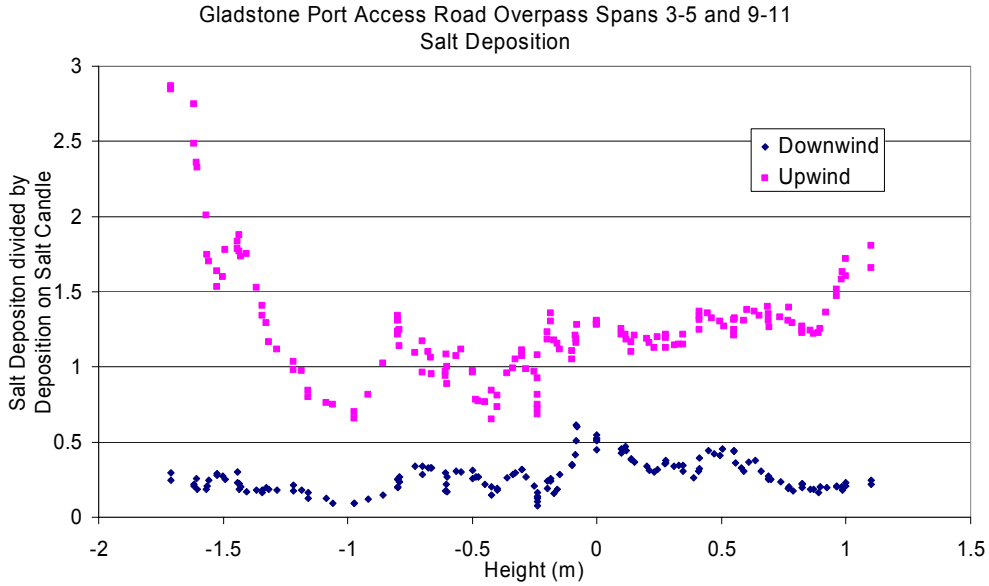
Figure 6.7b shows the salt deposition on the upwind face of the upwind parapet and the downwind face of the downwind parapet. That on the downwind (sheltered) face is about 0.25 DSC. That on the upwind (exposed) face is largest near the bottom, larger than average near the top, and roughly 1.1 DSC between the two.

To summarize the deposition on this superstructure, it is largest on upwind faces, intermediate on horizontal faces and least on downwind faces and in protected parts of the under bridge deck. The highest deposition rates of all are on the bottom edges of the two downwind girders and of the upwind face of the upwind parapet.

Figure 6.7 Salt deposition on the Gladstone Port Access Road overpass measured relative to the salt candle deposition



a)



b)

6.2.2 Stewart Road Overpass

The Stewart Road Overpass in Gold Coast City is located at latitude 28°8' and longitude 153°27'. It is on the Pacific Highway at the northern end of the new Tugun Bypass in Currumbin. There is ocean to the North, North-East and East of this bridge. The bridge has two spans of 21.8 metres. The superstructure consists of a reinforced concrete deck on 45 Type A prestressed concrete deck units. The units are laid with a nominal 35 mm gap. For the salt deposition simulation these deck units are taken together as a single unit but the gaps between them are used to together with concrete properties in estimating the aerodynamic roughness of the underside of the bridge. The superstructure is very wide with a total width of about 30.7 metres and the height is 2.3 metres, giving a height to width ratio of 1:13.3.

The height in the centre of the bridge deck is assumed to be 8.0 metres above ground level in the computer simulations of salt deposition.

The salt deposition on a salt candle was extracted from our GIS database at the location of the bridge for a marine environment at the latitude and longitude given, 38.9 $mg.m^{-2}.day^{-1}$. This does not take into account the bridge height.

The analysis of wind flow and salt deposition was carried out in a similar manner to the previous bridge and the results for the Stewart Road Overpass are shown in Figure 6.8.

Figure 6.8a shows the volume fraction of salt of aerosols that were released well upstream of the bridge, within 1.2 metres of the mid-height. The salt is transported first to the leading side of the parapet and much salt is deposited there. The flow separates from the top of the parapet, leaving an open recirculation region (shown in blue) behind it over the first barrier and the front portion of the bridge deck. The flow separating from the bottom edge of the parapet hits the lower portion of the first prestressed concrete deck unit. The salt-containing flow separates again from the bottom of this deck unit but reattaches a short distance along the underside of the deck.

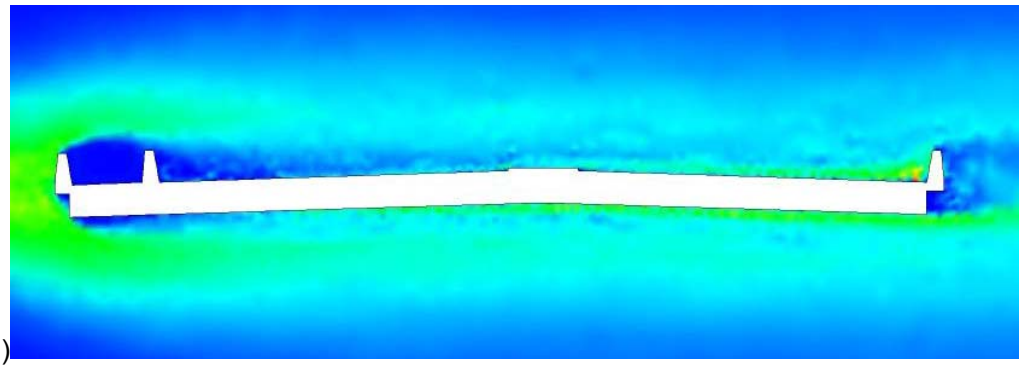
In Figure 6.8a, both the turbulence in the mean wind and that generated by the bridge itself slowly brings salt into the recirculation regions. Over and under the leeward side of the bridge deck, the salt moves into the boundary layer and builds up there to high concentrations. There is a second separation from the top of the leeward parapet and from the bottom of the last deck unit, leading to low salt concentrations downwind. The turbulence brings very little salt back onto the back of the downwind parapet.

Figure 6.8b shows the volume fraction of salt when aerosols were released between 1.2 and 2.4 metres of mid-height. There is very little difference between this and the top figure because turbulence upstream of the bridge has diffused the salt. There is less deposition on the upwind side of the parapet. More salt is trapped in the recirculation region over the front of the bridge deck. There is less salt deposition on the underside of the bridge deck.

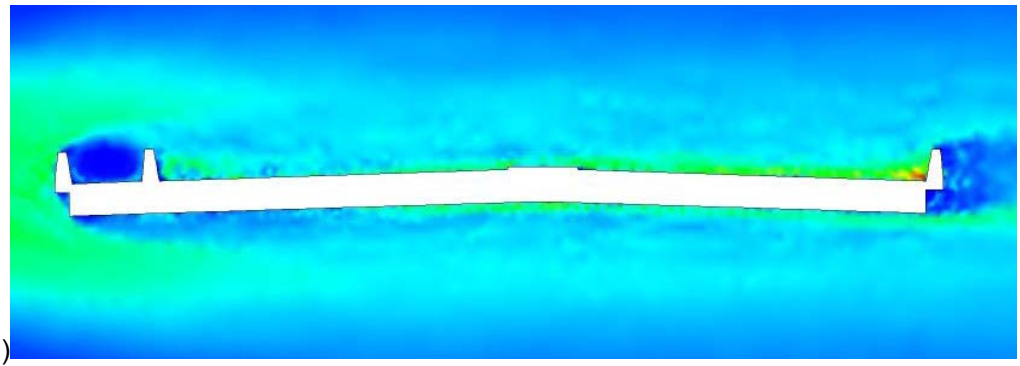
Figure 6.8c shows the sum of the top two figures plus salt released further from mid-height. Salt concentrations in the air are seen to be lowest between the leading parapet and the first barrier, downwind of the superstructure, under the front of the bridge deck and under the front parapet. Salt concentrations in the air are seen to be highest above the downwind end of the deck, behind the barrier above the deck, and under the downwind end of the deck. High salt concentrations in the air do not necessarily correspond to high deposition rates.

Figure 6.8 **Volume fraction of salt around the superstructure of the Stewart Road overpass; a) particles released within 1.4 m of the mid-height, b) particles released between 1.4m and 2.8m of mid-height, c) all salt aerosol particles. Flow is from left to right. Red is a high concentration of salt, blue is low concentration.**

a)



b)



c)

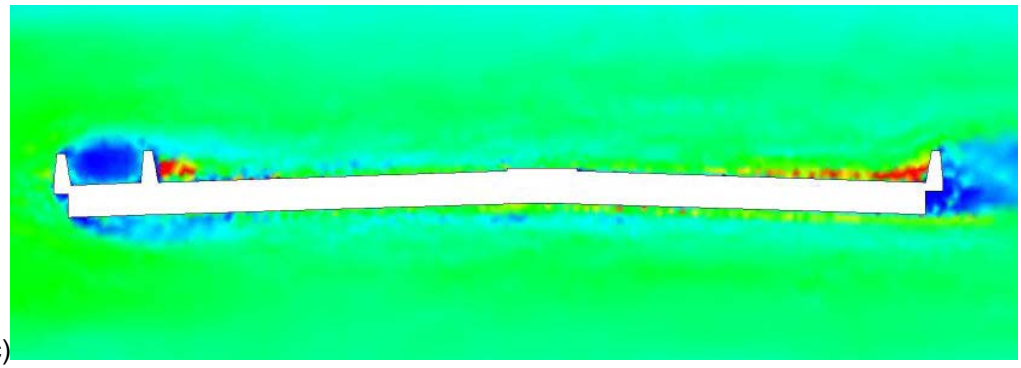


Figure 6.9 shows the locations of the zones used in analysing the deposition on the superstructure of the Stewart Road Overpass. This is referred to in the subsequent graphs of salt deposition and in Table 6.1.

Figure 6.9 **Locations of the zones for the superstructure of the Stewart Road overpass**

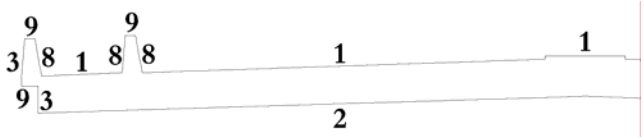


Figure 6.10 shows the salt deposition on the Stewart Road Overpass, measured relative to the salt candle deposition of 38.9 mg m^{-2} day^{-1} at the same location. The line marked “top” includes Zones 1 and 8, i.e. the top of the bridge deck, the inside of the parapets and both sides of the barrier. The deposition is highest in three places: on the inside face of the downwind parapet, on the upwind side of the median strip, and on the upwind side of the barrier. The deposition on the top of the bridge deck reaches a maximum of about 0.95 DSC ten metres upwind of the centre of the median strip. It is thus well upwind of the bridge centre; the high salt intensity in the air above the downwind end of the bridge deck (Figure 6.9a) does not lead to high deposition rates there.

The line marked “under” includes Zone 2, i.e. the underside of the bridge deck units. The greatest salt deposition is near the front edge of the upwind deck unit. The deposition on the underside of the deck is comparable in magnitude and position to that on the top of the deck.

Figure 6.10b shows the salt deposition on the upwind face of the upwind parapet and deck unit, and the downwind face of the downwind parapet and deck unit. That on the downwind (sheltered) face is about 0.2 DSC for the parapet and less for the deck unit. That on the upwind (exposed) face is largest near the bottom, larger than average near the top, roughly 1.5 DSC at the bottom of the parapet and smaller in the protected area at the top of the deck unit.

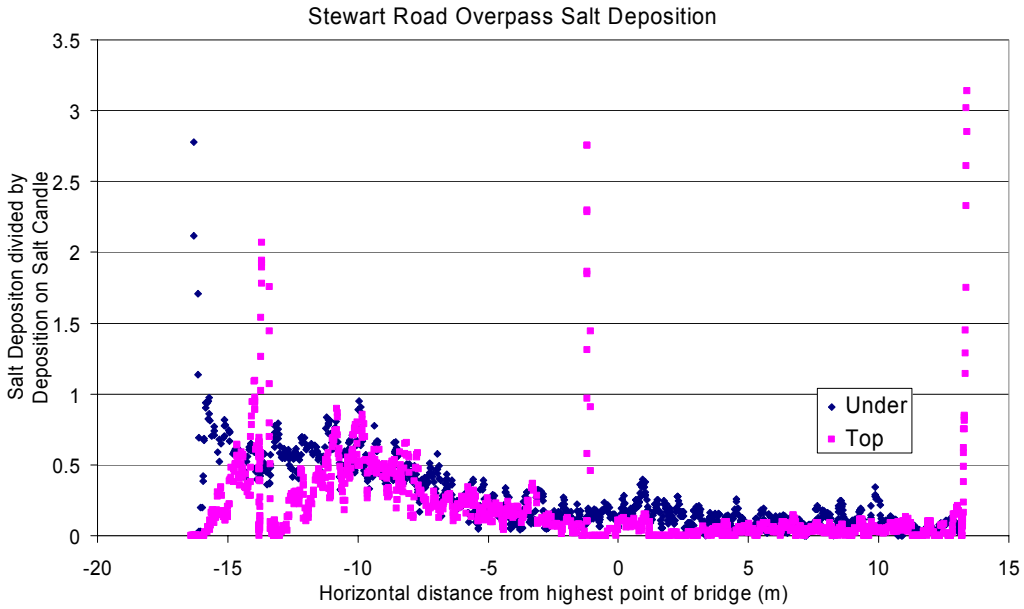
To summarize the deposition on this superstructure, it is largest on upwind faces including the upwind face of the median strip, and smallest in the downwind parts of the bridge deck, particularly on the back of the last deck unit. [No attempt was made to model the gaps between the deck units; it is expected that the salt deposition there will be very small].

6.2.3 South Johnstone River Bridge

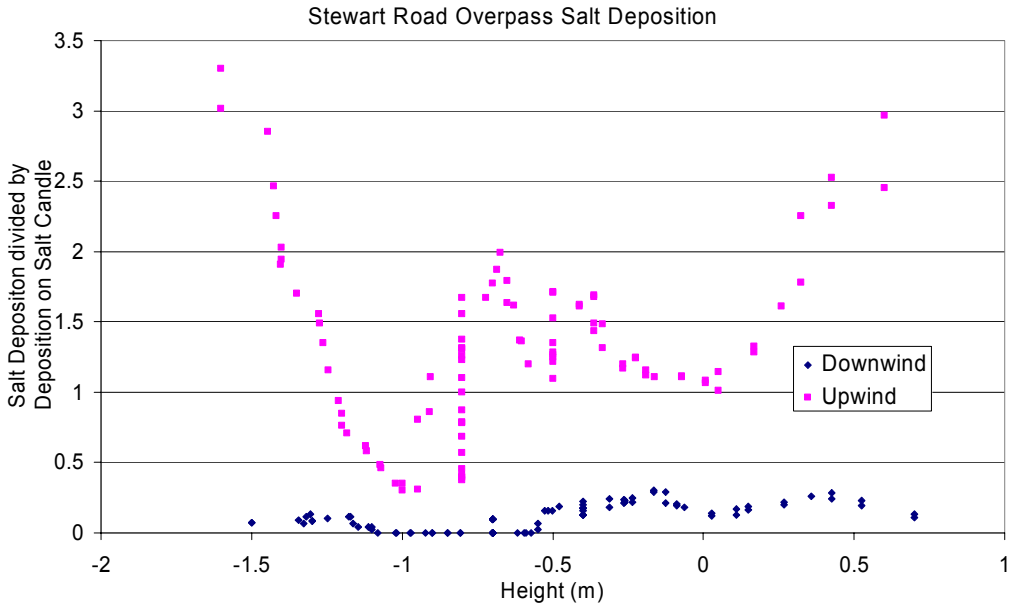
The South Johnstone River Bridge in the Johnstone Shire south of Innisfail is located at latitude 17°40’ and longitude 146°5’. It is on the Innisfail-Japoon Road. There is ocean to the East of this bridge.

The bridge has five spans, each about 25 metres long. The superstructure rests on rectangular prestressed concrete deck units. 17 units are spaced apart for spans 1 and 2 and support a reinforced concrete deck. 15 units are tied together for spans 3, 4 and 5. The bridge has a two-bar traffic rail on each side. The computer simulation of deposition is for the superstructure in spans 3, 4 and 5. This part of the superstructure has an overall width of 9.4 metres and the height including the two bars of the traffic rail is 1.65 metres, giving a height to width ratio of 1:5.7.

Figure 6.10 Salt deposition on the Stewart Road Overpass measured relative to the salt candle deposition



a)



b)

The height in the centre of the bridge deck is assumed to be 10.9 metres above water level in the computer simulations of salt deposition. This is consistent with the height of the centre span.

The salt deposition on a salt candle was extracted from our GIS database at the latitude and longitude given. If this is taken as a “marine” environment then the salt deposition is 10.0 mg m^{-2} day^{-1}. For a non-marine environment it is 6.7 mg m^{-2} day^{-1}. The deposition on the salt candle for this bridge was also computed.

The results of the salt deposition analysis for the South Johnstone River Bridge are illustrated in Figure 6.11.

Figure 6.11a shows the volume fraction of salt of aerosols that were released just upstream of the bridge, within 1.2 metres of the mid-height. The salt is transported first to the leading side of the deck and much salt is deposited there. A lot of salt is also deposited on the front sides of the two upwind traffic rails. There is little flow separation from the bottom of the upwind deck unit, from the top of the deck and behind the traffic rails. The top and the bottom of the deck are slightly sheltered. The downwind traffic rails are collecting some salt. Turbulence brings a little salt back onto the downwind end of the deck.

There are clear differences between the top and middle figures. Figure 6.11b shows the volume fraction of salt when aerosols were released between 1.2 and 2.4 metres of mid-height. This salt misses the upwind end of the deck and upwind traffic rails almost completely, although some is deposited on the downwind half of the deck and on the downwind traffic rails. Turbulence brings more salt back onto the downwind end of the deck.

Figure 6.11c shows the sum of the top two figures plus salt released further from mid-height. Salt concentrations in the air are seen to be lowest above the deck, below the leading third of the deck, behind the traffic rails, and downwind of the bridge. Salt concentrations in the air are seen to be highest on the upwind side of the deck and in front of and above the upwind traffic rails.

Figure 6.12 shows the locations of the zones used in analysing the deposition on the superstructure of the Stewart Road Overpass. This is referred to in the subsequent graphs of salt deposition and in Table 6.1.

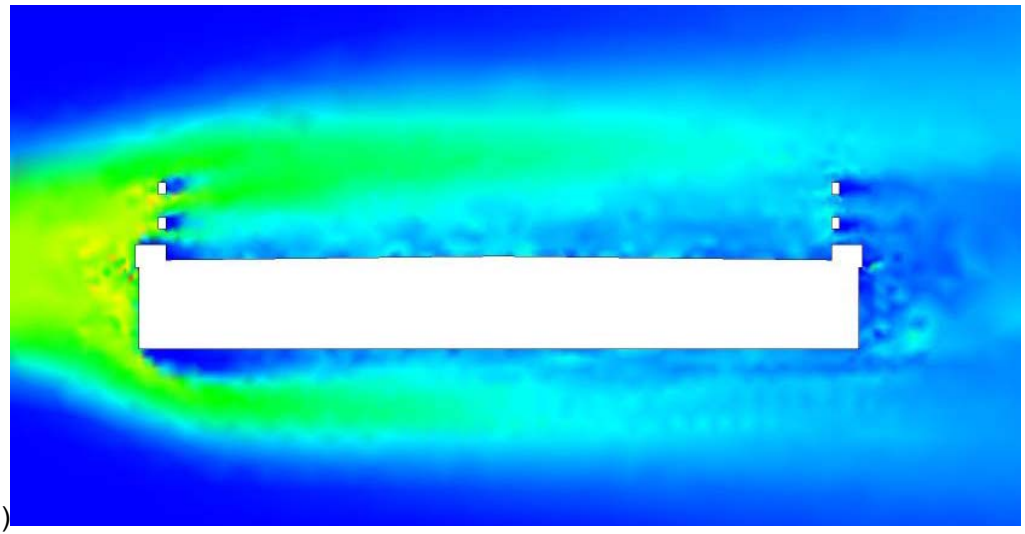
Figure 6.13a shows the salt deposition on superstructure of the South Johnstone River Bridge, measured relative to the salt candle deposition of 6.7 or 10 mg m-2 day-1 (see above for details) at the same location. The deposition on the traffic rails is not included in the figure, the average over all traffic rails is a very high 2.53 DSC. The line marked “top” includes Zones 1 and 8, i.e. the top of the bridge deck and kerbs. The deposition is fairly uniform, averaging about 0.9 DSC over the downwind half, with a small spike on the downwind kerb.

The line marked “under” includes Zone 2, i.e. the underside of the bridge deck units. The greatest salt deposition is near the front edge of the upwind deck unit this then drops off rapidly to zero before increasing to a fairly uniform value of about 1.1 DSC.

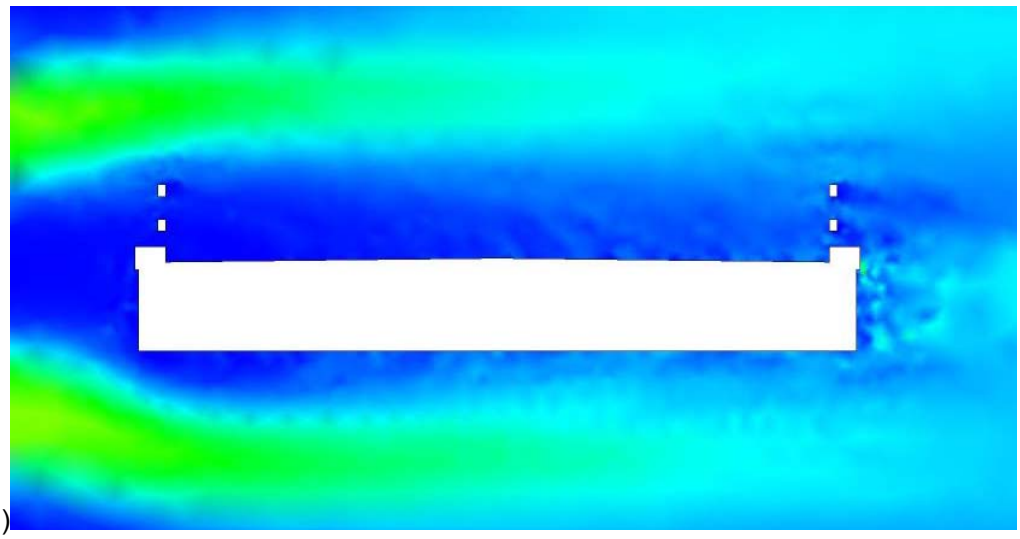
Figure 6.13b shows the salt deposition on the upwind and downwind sides of the deck. There is an unimportant discontinuity at height zero at the boundary between the deck unit and the reinforced concrete above. That on the downwind face is remarkably high, peaking at 1.0 DSC on the deck unit. That on the upwind (exposed) face is largest near the top, going up to approximately four times the salt candle. At the bottom it climbs to three times. In the middle it averages something like one times.

Figure 6.11 Volume fraction of salt around the superstructure of the South Johnstone River Bridge; a) particles released within 1.4 metres of the mid-height, b) particles were released between 1.4 and 2.8 metres of mid-height, c) all salt aerosol particles.

a)



b)



c)

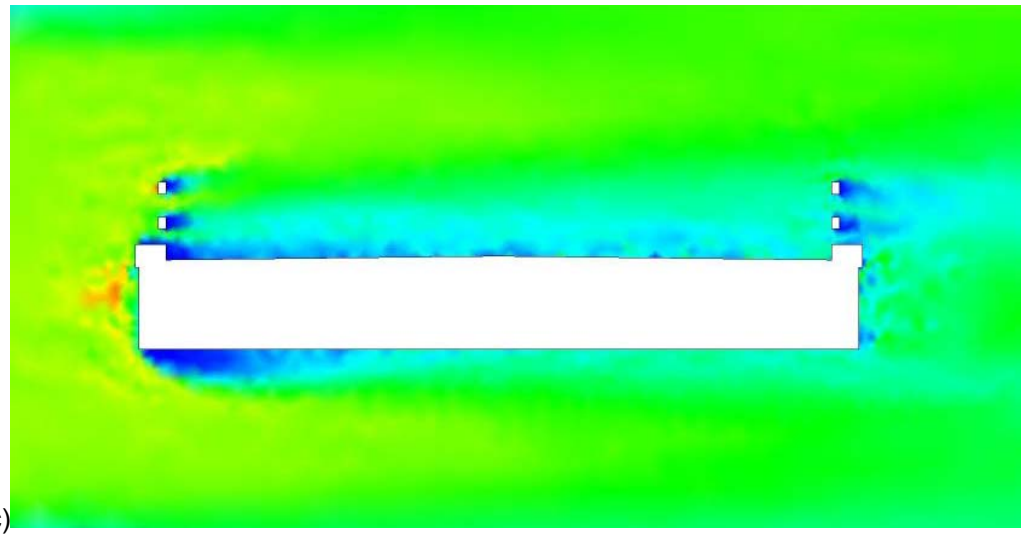


Figure 6.12 Locations of the zones for the superstructure of South Johnstone River Bridge

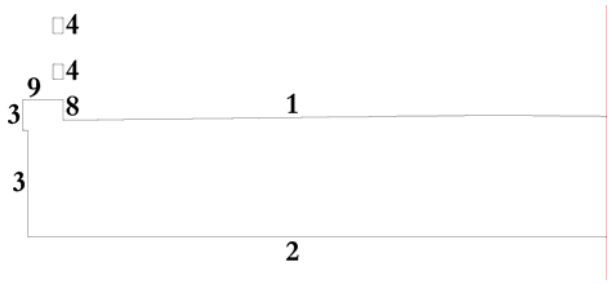
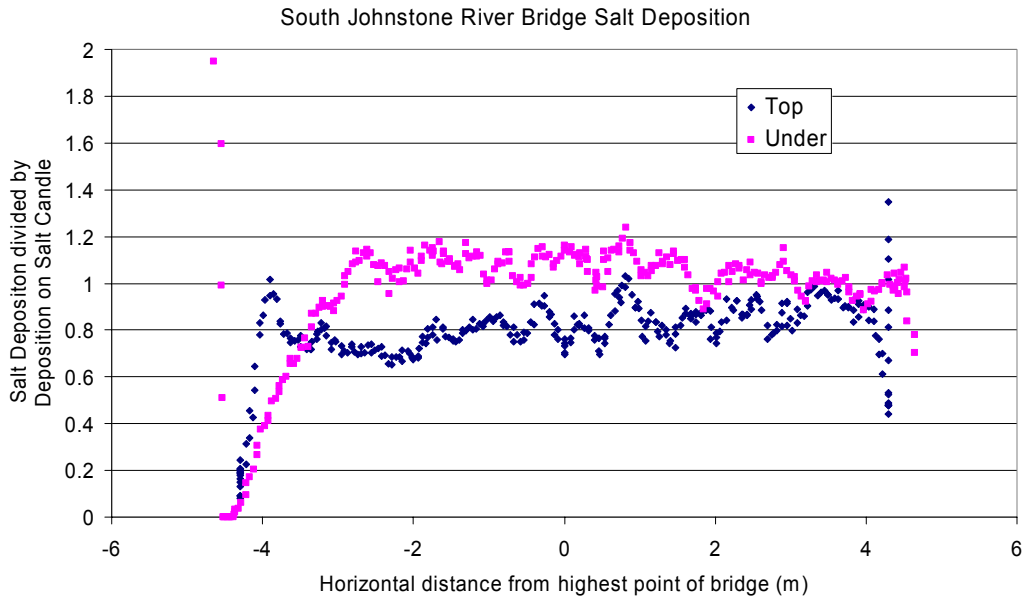
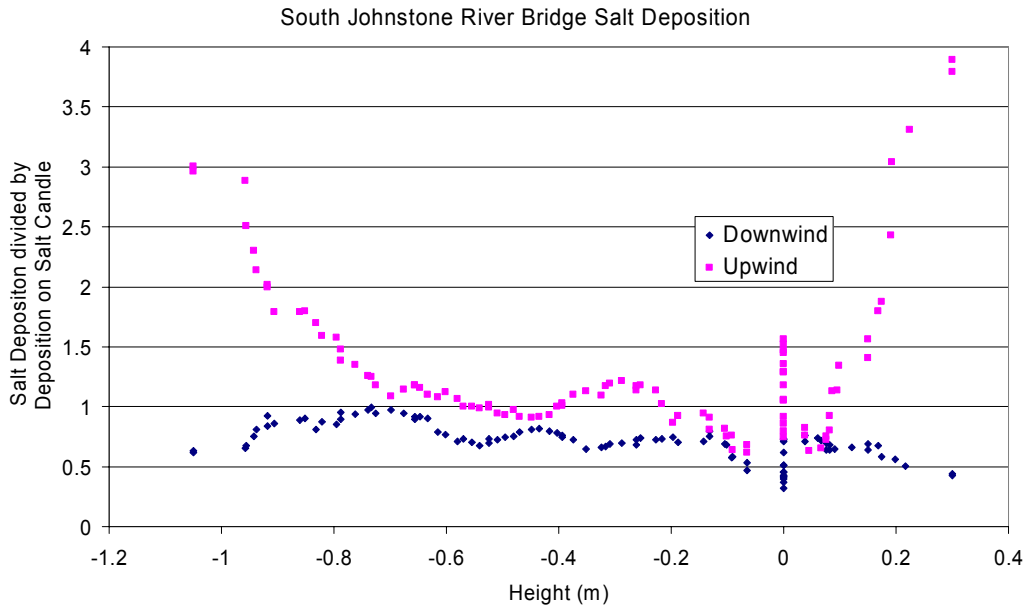


Figure 6.13 Salt deposition relative to that on a salt candle for the South Johnstone River Bridge





To summarize the deposition on this superstructure, it is particularly large on the bottom and top of the upwind face, going up to about four times the salt candle. The average deposition on the traffic rails is also large, about two and a half times that on a salt candle. The deposition on the top of the deck is slightly larger than that on the bottom and there is a remarkably high deposition rate on the downwind side of the deck.

6.2.4 Johnson Creek Bridge

The Johnson Creek Bridge in Mount Isa City is located at latitude 20°40' and longitude 139°25'. It is on the Barkly Highway between Mt Isa and Camooweal. It is well inland and not near any ocean. The bridge has three uneven spans, the centre span is the longest at 18 metres long. The superstructure consists of a reinforced concrete deck on rectangular prestressed concrete deck units. There are 15 deck units, placed with a nominal 25 mm gaps. The bridge has a two-bar traffic rail on each side. The superstructure has an overall width of 10 metres and the height including the two bars of the traffic rail is 1.37 metres, giving a height to width ratio of 1:7.3.

The height in the centre of the bridge deck is assumed to be 5.6 metres above ground level in the computer simulations of salt deposition.

The salt deposition on a salt candle was extracted from our GIS database at the location of the bridge at latitude and longitude given. It is 3.8mg.m-2.day^{-1}. The deposition on the salt candle for this bridge was also computed.

Concentrations of salt in the air for the Johnson Creek Bridge are shown in Figure 6.14.

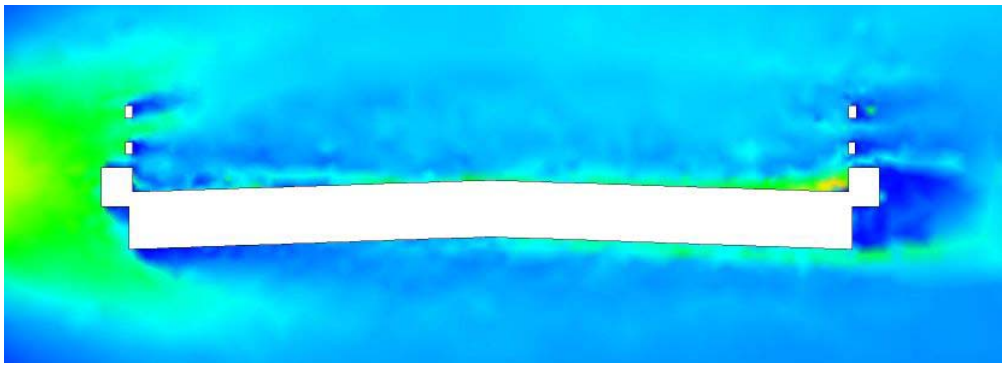
Figure 6.14a shows the volume fraction of salt of aerosols that were released upstream of the bridge, within one metre of the mid-height. The salt is transported first to the leading side of the deck and much salt is deposited there. A lot of salt is also deposited on the front sides of the two upwind traffic rails. Flow separation from the bottom of the upwind side of the deck affects deposition on the upwind concrete deck unit. There is a little flow separation from the bottom of the upwind deck unit, from the top of the deck and behind the traffic rails. The top and the bottom of the deck are slightly sheltered, but turbulence brings salt back into contact with the downwind half. The downwind traffic rails are collecting some salt. Turbulence brings some salt back onto the downwind end of the deck.

Figure 6.14b shows the volume fraction of salt when aerosols were released between one and two metres of mid-height. There is not much difference between this and Figure 6.14a. There is less deposition on the upwind side top and bottom of the deck.

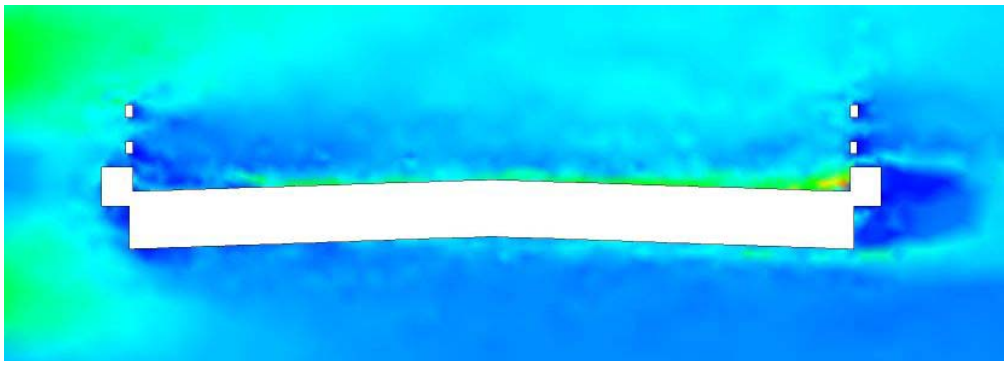
Figure 6.14c shows the sum of the top two figures plus salt released further from mid-height. Salt concentrations in the air are seen to be lowest above the upwind end of the deck, below the leading end of the deck, behind the traffic rails, and downwind of the bridge. Salt concentrations in the air are seen to be highest above and below the downwind ends of the deck.

Figure 6.14 Volume fraction of salt around the superstructure of the Johnson Creek Bridge; a) particles released within 1.4 m of the mid-height, b) particles released between 1.4m and 2.8m of mid-height, c) all salt aerosol particles. Flow is from left to right.

a)



b)



c)

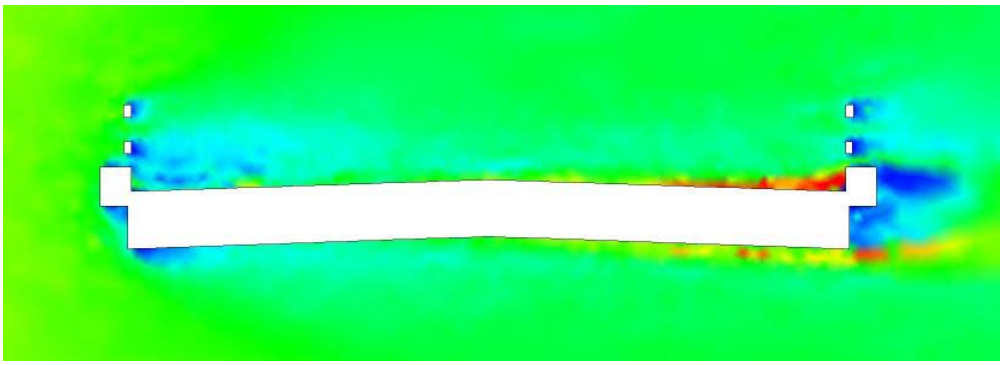


Figure 6.15 shows the locations of the zones used in analysing the deposition on the superstructure of the Stewart Road Overpass. This is referred to in the subsequent graphs of salt deposition and in the table from Section 3.

Figure 6.15 Location of zones for the superstructure of the Johnson Creek Bridge

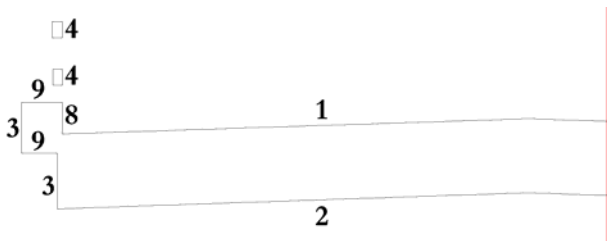


Figure 6.16 shows the salt deposition on the superstructure of the Johnson Creek Bridge, measured relative to the salt candle deposition of 3.8 mg m^{-2} day^{-1}. The deposition is seen to be intermediate in distribution between that on the Stewart Road Overpass (in which deposition drops to near zero at the downwind end of the deck) and on the South Johnstone River Bridge (where the deposition is nearly constant. This is because the height to width ratio of the Johnson Creek Bridge deck is 1:7.3, which is between those of the other two bridges (1:13.3 and 1:5.7).

The line marked “top” includes Zones 1 and 8, i.e. the top of the bridge deck and kerbs. The deposition decays from a maximum of 0.6 DSC to a minimum of about 0.2 DSC, with a large spike on the downwind kerb.

The line marked “under” includes Zone 2, i.e. the underside of the bridge deck units. The greatest salt deposition is near the front edge of the upwind deck unit this then drops off and rapidly rises again to a maximum of 1.2 DSC before reducing slowly to a minimum of about 0.2 DSC.

Figure 6.16b shows the salt deposition on the upwind face and deck unit, and the downwind face and deck unit. The pattern of deposition closely resembles at of the Stewart Road Overpass, because the geometry is similar. The deposition on the downwind (sheltered) face is about 0.15 DSC for the parapet and less for the deck unit. That on the upwind (exposed) face is largest near the top at about 2.8 DSC, larger than average near the bottom at 2.35 DSC, roughly 1.8 DSC at the bottom of the upwind face and smaller in the protected area at the top of the leading deck unit.

6.2.5 Bridge over Ward River

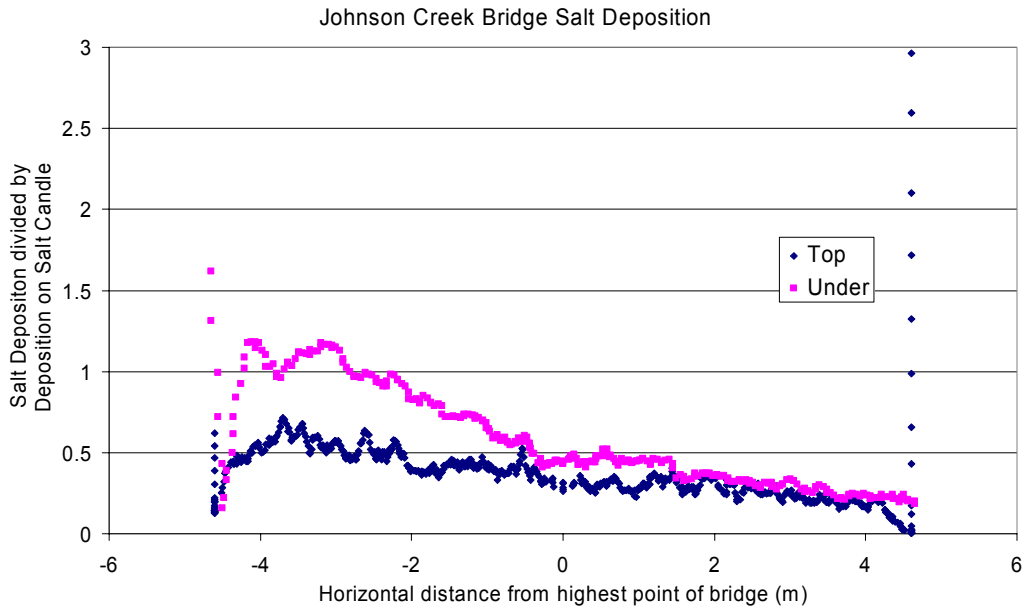
The bridge over the Ward River in Murweh Shire is located at latitude 26°30’ and longitude 146°15’. It is on the Diamantina Developmental Road between Charleville and Quilpie.

The bridge has six spans, each about 13.6 metres long. The superstructure consists of a reinforced concrete deck on two steel I-beams. It has a total width of 4.27 metres and the height including girders is 1.05 metres, giving a height to width ratio of 1:4.1.

The salt deposition on a salt candle was extracted from our GIS database at the latitude and longitude given. It is 3.9 $mg.m^{-2}.day^{-1}$. The deposition on the salt candle for this bridge was also computed.

Concentrations of salt in the air for the bridge over the Ward River are shown in Figure 6.17. The distribution of salt in the air is remarkably symmetric; it is not immediately obvious that the wind from is from left to right. As on the other bridges, there is a reasonable amount of deposition on the upwind face and flow separation above and below, but this time the salt deposition on the bottom of the front of the leading girder is not very large.

Figure 6.16 Salt deposition relative to that on a salt candle for the Johnson Creek Bridge



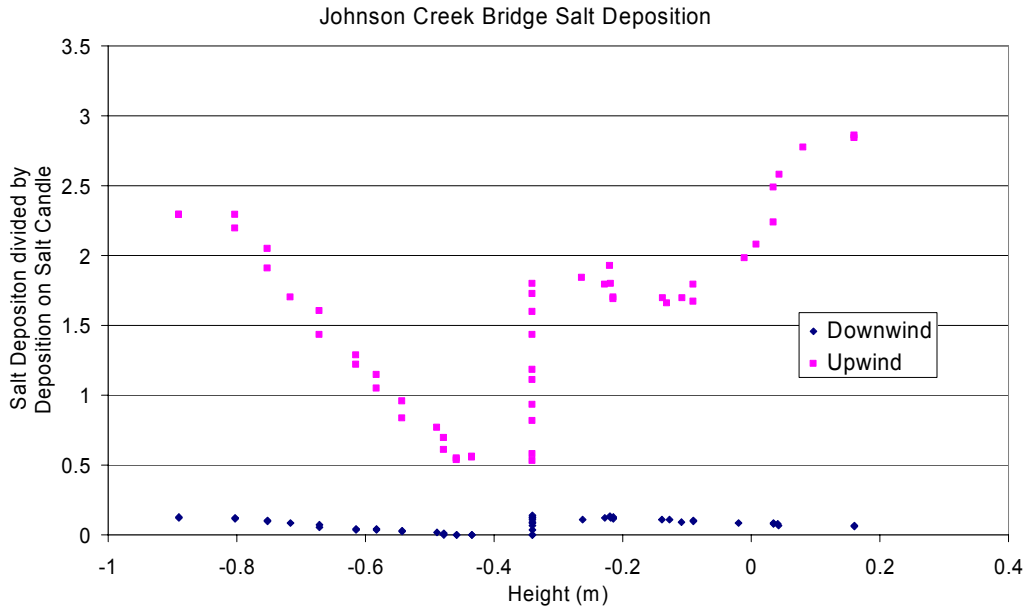
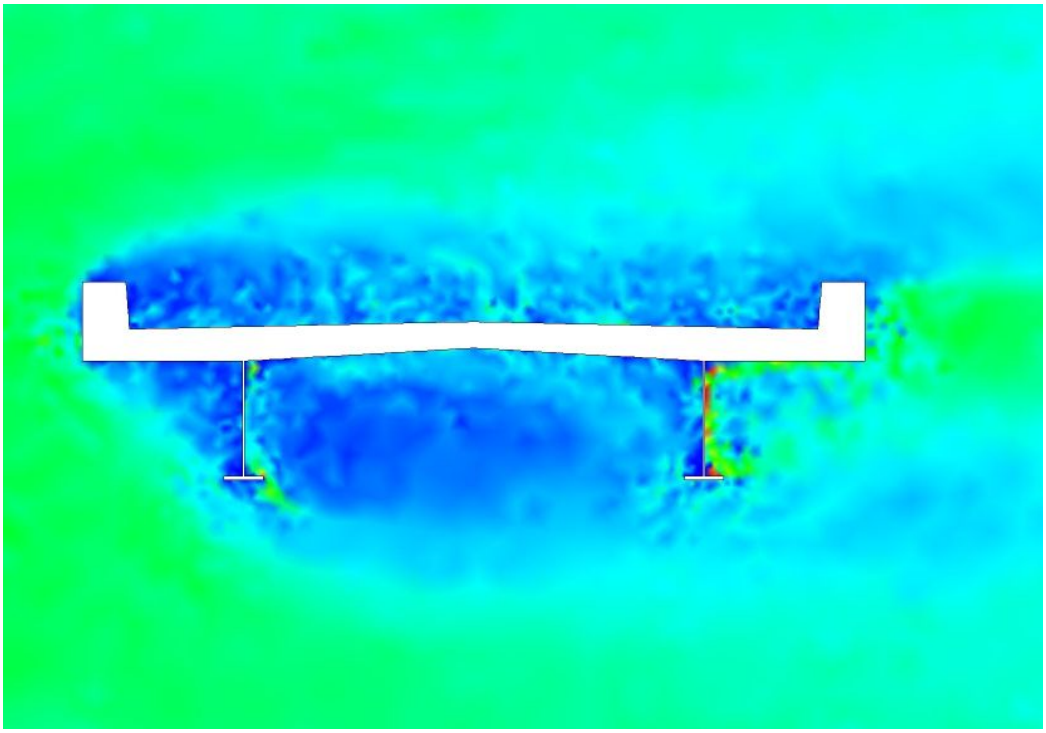


Figure 6.17 Distribution of salt for the Ward River Bridge



There is a big recirculation zone between the girders and salt hangs around in the low velocity regions behind both girders. The superstructure height to width ratio is large enough (1:4.1) to bring large amounts of salt into the area downwind of the bridge. The region above the deck behaves as expected, salt is brought down to the road surface by turbulence.

Figure 6.18 shows the locations of the zones used in analysing the deposition on the superstructure of the Stewart Road Overpass. This is referred to Table 6.1.

Figure 6.18 Locations of zones for the Ward River Bridge

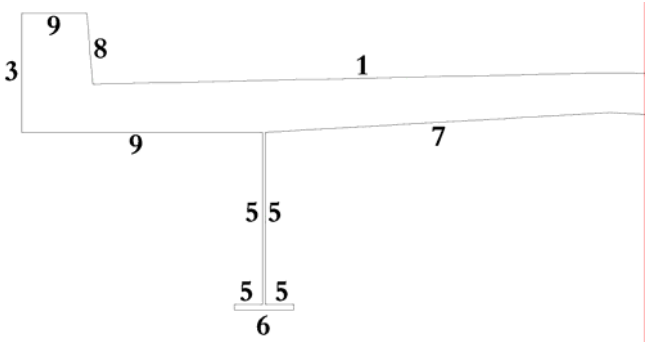
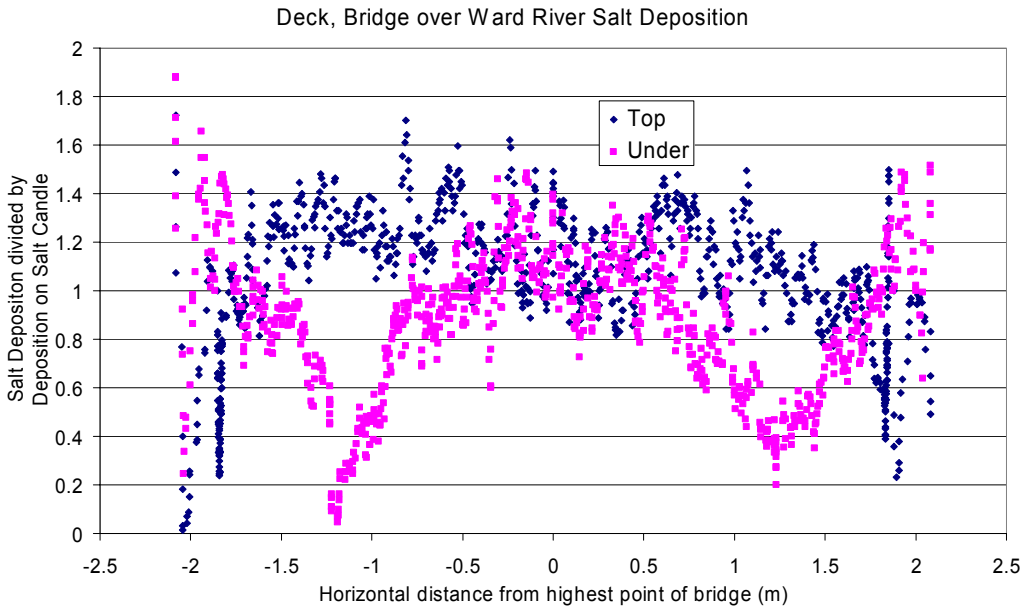
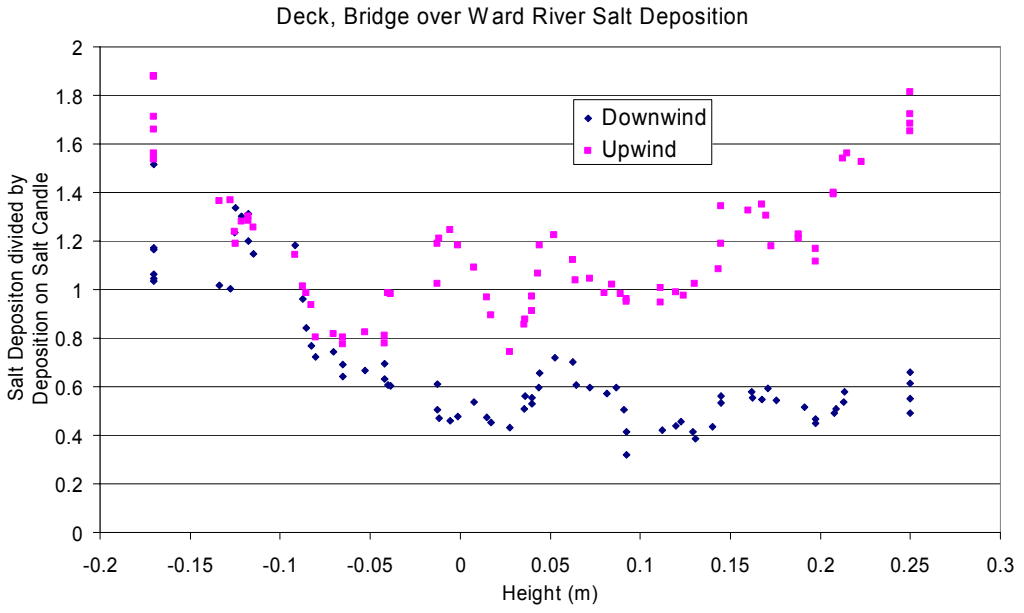


Figure 6.19 shows the salt deposition on the deck, kerbs and the top flanges of the steel girders of the bridge over the Ward River, measured relative to the salt candle deposition of 3.9 mg m^{-2} day^{-1}. As for the the South Johnson River Bridge, the salt deposition on the top of the deck is fairly uniform, but larger here at about 1.3 DSC for much of the deck. The salt deposition under the deck is not too different from that above, it is sheltered near where the girders intersect the deck, but not midway between the girders or further downwind.

Figure 6.19 Salt deposition relative to that on salt candle for the deck of the Ward River Bridge

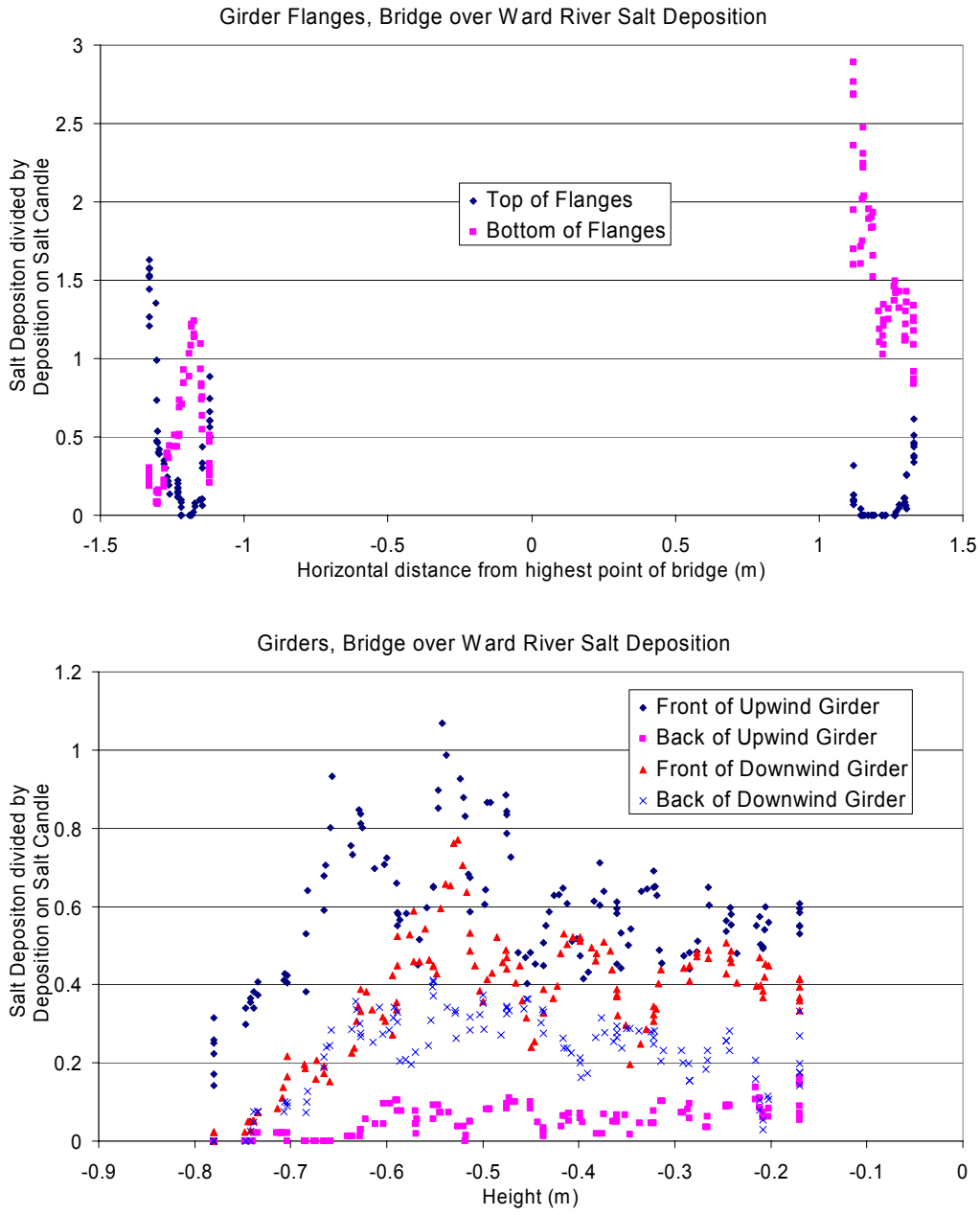




As before, the salt deposition on the upwind face of the deck is largest at the top and bottom, with peak values of about 1.9 DSC and reducing to about 1.0 DSC at mid height. The salt deposition on the downwind face of the deck is remarkably large, ranging from an average of about 0.5 DSC over the upper half to a maximum of about 1.5 DSC at the lower edge.

Figure 6.20 shows the salt deposition on the webs and lower flanges of the steel girders of the bridge over the Ward River, measured relative to the salt candle deposition. The bottom of the flange of the downwind girder has the largest salt deposition of the whole bridge, rising to about a peak of about 2.9 DSC. This is the reattachment zone for the recirculation region separating from the upwind girder, and has a high local turbulence that aids deposition. The top of this flange, by way of contrast, has a low deposition rate.

Figure 6.20 Salt deposition relative to that on salt candle for the girders of the Ward River Bridge



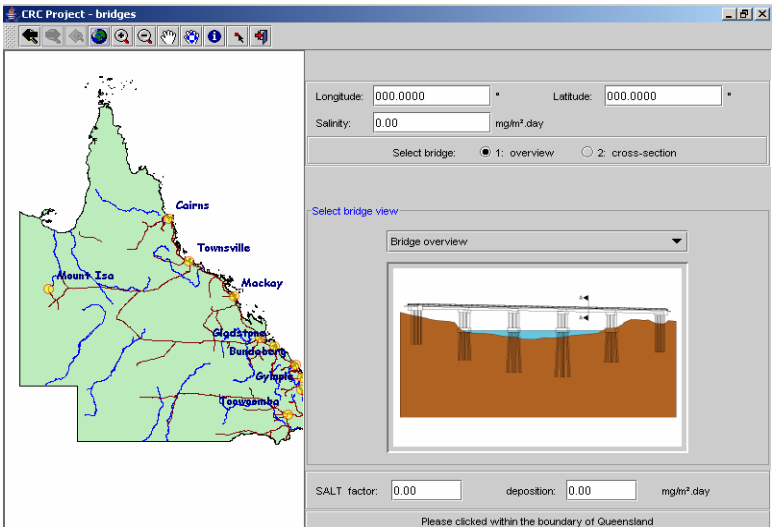
The leading edge of the top of the flange of the upwind girder has a high deposition rate, peaking at about 1.7 DSC. This is expected as it catches the salt coming off the front of the bridge.

For the webs of the girders, the front of the upwind girder has the largest salt deposition rate and the back of that girder has the smallest. None have a particularly large deposition rate, with only one point above 1.0 DSC.

6.3 Program and User Interface

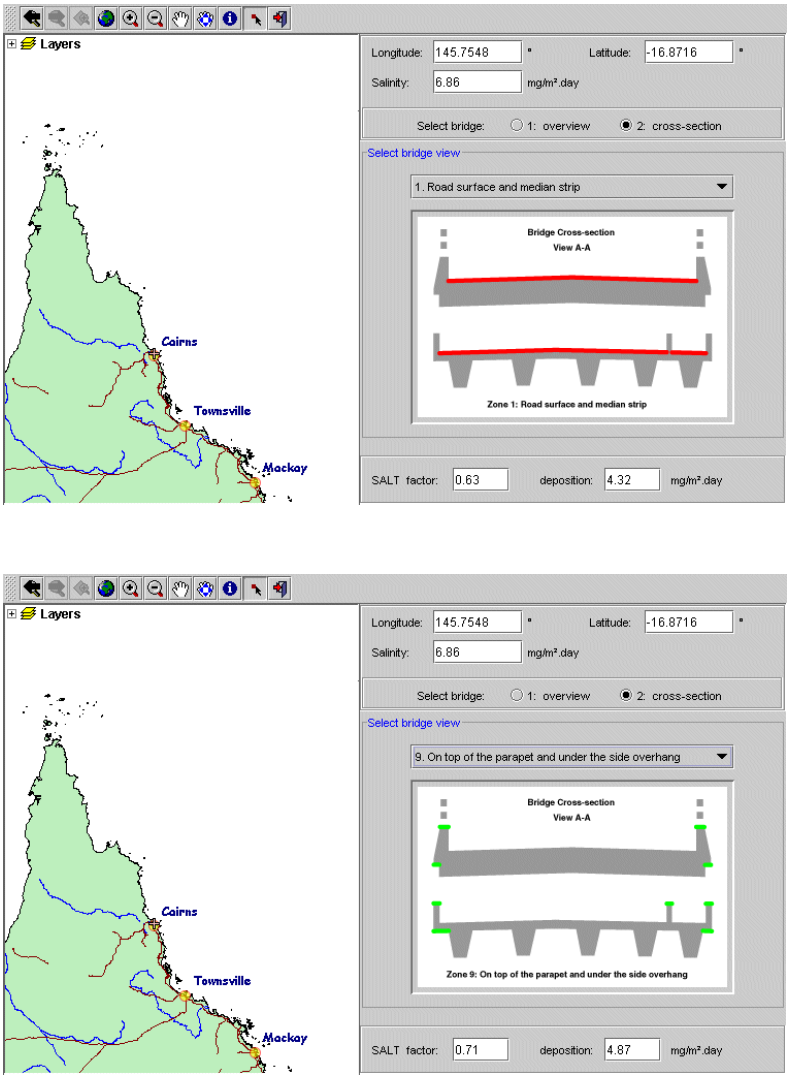
The information derived from the analysis of the five bridges was amalgamated to give a generic bridge structure with nine different zones as shown in Figure 6.1. Salt factors were derived for each zone to modify the salt deposition levels. A GUI has been designed and implemented incorporating the GIS of Queensland such that clicking on a point on the map will get the salt deposition for that point. (Figure 6.21) One of the nine bridge zones can then be selected and the expected salt deposition on that zone will be calculated from the GIS figure and the salt factor for the zone.

Figure 6.21 Initial Screen in Bridge program allowing selection of point in Queensland



Two examples of the bridge zones being selected in the GUI are illustrated in Figure 6.22. (The zoom facility in the GIS map is also illustrated).

Figure 6.22 Two frames of the GUI showing different bridge zones selected



6.4 Utility of Present Results

These results represent the first step towards developing a CBR program for life prediction of metallic bridge elements. Cases for two typical bridge types have been derived from CFD modelling of salt deposition. The salt values provided by the program have not been verified against actual deposition on the bridges (there was no provision for verification in the project program).

7. SITE VISIT

As part of this project a trip was undertaken to look at corrosion concerns of the industrial partners in the context of the software tools being developed. David Paterson and Wayne Ganther from CSIRO travelled to the Sunshine Coast with Alan Carse of Queensland Department of Main Roads and Michael Ball of Queensland Department of Public Works. They were joined for part of the visits by Ed Bowers of QBuild which is the commercial unit of Public Works responsible for maintenance of the buildings and infrastructure supported by the department. The Sunshine Coast area was chosen for the visit due its coastal location and known corrosion problems.

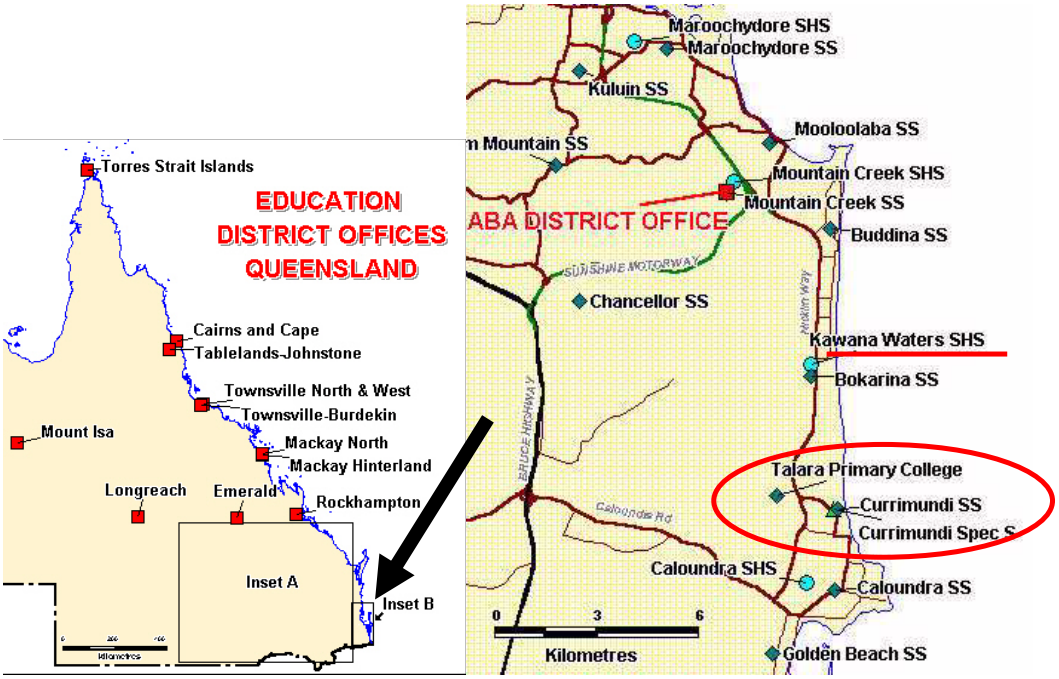
A detailed summary of the visit is given in two reports prepared by Wayne Ganther:

- Trip to Sunshine Coast Queensland, September 2004, Visit to Schools, Report No 2002-059-B No 7, and,
- Trip to Sunshine Coast Queensland, September 2004, Visit to Bridge and Foreshore, Report No 2002-059-B No 8.

7.1 Visit to Schools

Four schools were visited on the Sunshine Coast in the area shown by the maps in Figure 7.1.

Figure 7.1 Maps showing the location of the schools visited



The four schools were:

- Currimundi State School, located within a couple of hundred metres of Dicky Beach, The school opened in 1977 and the buildings are between 4 and 12 years old,

- Currimundi Special School, located across the road from the State School. The school was opened in 1984.
- Talara Primary College, located approximately 2 kilometres from the coast (Dicky Beach) almost directly west of the Currimundi schools. The school was opened in 1998 and Block E was only opened in 2004. and
- Kawana Waters State High School, located approximately 1 kilometre from the coast and approximately 4 kilometres north of Currimundi. The school was opened in 1986.

All of the schools have significant corrosion problems. Most of the corrosion issues relate to sheltered corrosion and have been seen in similar structures in Victoria. The main structures affected are covered walkways and shelters ie. all areas where salt can be deposited and is not washed away by rainfall. Another problem is roof fasteners which have corroded; this may be due to inappropriate specifications as some fasteners were stainless steel and performing well. Other corrosion problems are due to inappropriate design, specifications or building practice. Some of the problems identified are illustrated in Figure 7.2 to Figure 7.9.

Figure 7.2 Rusting and deterioration at joins of gutters at Currimundi State School

Figure 7.3 Roof fasteners showing evidence of rust at Currimundi State School

Figure 7.4 Contact between stainless steel strapping and Colorbond® roof is causing deterioration of Colorbond®. Strapping not in contact is showing considerable corrosion (Currimundi Special School)

Figure 7.5 Triple grips and bolts on covered setdown showing evidence of red rust at Currimundi Special School

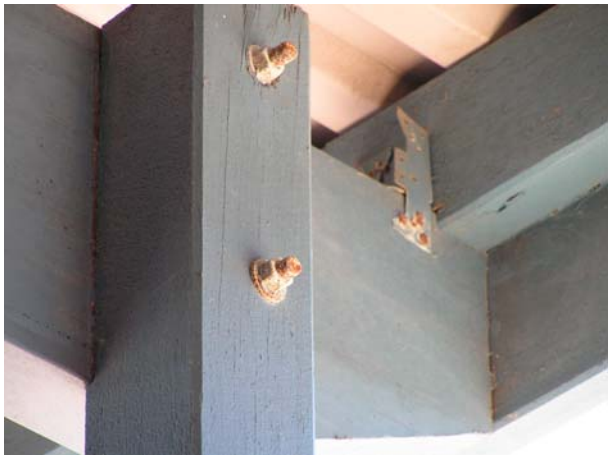


Figure 7.6 Fasteners in sheeting under porch of Administration block at Talara Primary College

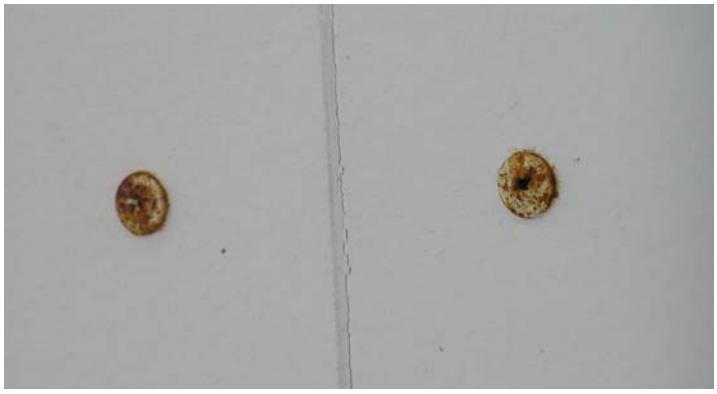


Figure 7.7 Degradation of gutter at join to drainpipe at Talara Primary College. Pop rivets have corroded away.

Figure 7.8 Underside of aluminium roof sheeting of covered walkway at Kawana Waters State High School

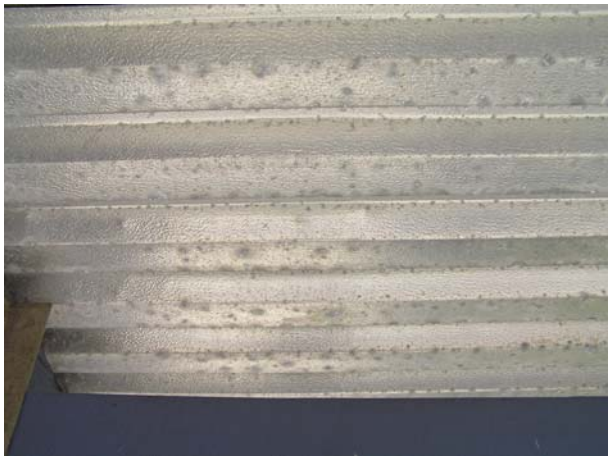


Figure 7.9 Heavily corroded fastener in walkway at Kawana Waters State High School

7.2 Bridge and Foreshore Visit

The project group also visited a bridge on the David Low Way (Figure 7.10) at Sunrise Beach near Noosa. This bridge was in a severe marine environment (Figure 7.11) with high salt content in the concrete and corrosion of the galvanised handrails and barriers.

Figure 7.10 Bridge on the David Low Way

Figure 7.11 View from the bridge showing proximity to the beach

Some of the corrosion problems identified on the bridge structure are illustrated in Figure 7.12 and Figure 7.13.

Figure 7.12 Corrosion on support beam of bridge

Figure 7.13 White corrosion product on bridge railing

An area on the foreshore near Noosa was also visited to have a look at how severe the coast was in terms of corrosion. Structures along the coast were inspected to see how they were fairing in the environment. It was found that some, if not most, of the infrastructure installed along the coast was incorrectly specified. Some of the components used were not suitable for the severity of the environment eg. the supports for the shade umbrellas were painted steel which was severely corroded after a short exposure (Figure 7.14). Where more resistant metals were specified they were not fully specified. The stainless steel handrails and plaques were a case in point where the level of finish would seem to be not correctly specified. The stainless steel had excessive "tea staining" (Figure 7.15) which would have been avoided if electropolishing had been specified.

Figure 7.14 Umbrella supports showing severe corrosion

Figure 7.15 Plaque showing "tea staining" from corrosion

8. FUTURE DIRECTIONS

This project has scoped out the applicability of a case-based reasoning paradigm for a software tool for lifetime prediction of metallic building components. Two applications have been developed, one more advanced than the other. In the application for QDPW, the focus was gutters in Queensland schools. A CBR engine has been designed and parts of it have been implemented in conjunction with the development of relevant databases of component life. The CSIRO holistic model has been modified to include the materials of relevance to gutters, including Colorbond®. In the application for QDMR the focus was metallic elements of bridges in Queensland. This required analysis of bridge structures to define structural elements in common that would be used as cases in a case-based situation and has been used to predict salinity levels for these cases.

8.1 CBR Engine

These software applications require further development to generate a commercially usable product. The design of the CBR engine is such as to allow the development of a comprehensive tool that can span a wide range of materials and a variety of environments, covering buildings, constructed facilities and infrastructure. The current tools have been developed as proof of concept with a very limited field of application.

Some modification of the case-based reasoning program will also be necessary to fully implement an inference engine and optimise the selection of cases and construct the final case input values from the alternatives retrieved from the databases. At present, the CBR can interrogate the various databases and select cases considered to be relevant to a given situation. There is no process for selecting which of the retrieved information should be stored as a new case.

8.2 Building Applications

The QDPW application could be extended into consideration of the whole building façade including roofs, gutters, drainpipes, windows and other metallic components. Air conditioning components also constitute an area where significant corrosion is an issue for facility managers and inclusion of this could provide benefits. As was done for gutters, cleaning models would need to be developed for different components, utilising CFD and water flow analysis, to facilitate modification of the salt deposition levels in the holistic model.

One important aspect in any program extension would be to ensure that nomenclature of components was consistent with other areas of design and Life Cycle Analysis so that the final tool could provide input to these processes. LCA workers use the definitions in the Australian Cost Management Manual.

These tools are dependent on the integrity of the data in the various databases. Information based on maintenance is of high importance in that it gives real data on the lifetime of components in actual situations. (In contrast to expert's opinions in the Delphi survey or results from modelling) A limited amount of data was received from QDPW and entered into a "Maintenance" database. It would be very useful to have some means of updating this database to reflect the knowledge accumulation as maintenance proceeds. As the system is currently designed, only the casebase is capable of being updated with new cases, derived from the databases already included, and there is no interface for ongoing input of new maintenance data. To increase the utility of the programs then methods need to be found for ongoing updating of, particularly, the maintenance database.

8.3 Bridge Application

The work done to date has not linked the bridge information into the CBR engine. Initial analysis has been carried out to identify the structural elements for cases and the current environmental setting can be extended lifetime prediction and risk assessment, with the development of relevant databases and incorporation into the CBR engine. Two other areas of interest to QDMR are below ground corrosion modelling and the extension of the program to include other materials, in particular concrete.

The lifetime prediction tool for bridges will be developed by forming databases of maintenance information and the model data from the CFD analysis. Parameters relevant to below-ground metal corrosion will be identified and the model adapted accordingly. For extension to concrete, the existing CSIRO and international work on modelling concrete degradation will be reviewed to determine the most appropriate algorithms.

9. REFERENCES

Baghni, I.M., Lyon, S.B., Ding, B., 2004. The effect of strontium and chromate ions on the inhibition of zinc. Surface & Coatings Technology, 185, 194-198.

Bauer, D.R., 2000. Global exposure models for automotive coating photo-oxidation. Polymer Degradation and Stability, 69, 297-306.

Bauer, D.R., 2000. Interpreting weathering acceleration factors for automotive coatings using exposure models. Polymer Degradation and Stability, 69, 307-316.

Bluescope Steel, 2005. Colorbond steel data sheet: Exterior roofing and walling. www.bluescope.com.au.

Boothroyd,G. 1994, Product design for manufacture and assembly, *Computer-Aided Design*, 26(7), 505-519.

Cole, I.S. "Recent Progress in Modelling Atmospheric Corrosion", *Corrosion Reviews*, Volume 20, Nos. 4-5, 2002, Editor M. Schorr.

Cole, I.S., Lau, D. and Paterson, D.A, 2004, Holistic model for atmospheric corrosion – Part 6- From wet aerosol to salt deposit. *Corros End Sci Techn* 39(3): 209-218.

Cole, I.S. and Paterson, D.A., 2004, Holistic model for atmospheric corrosion – Part 5 – Factors controlling deposition of salt aerosol on candles, plates and buildings. *Corros Eng Sci Techn* ,39(2): 125-130.

Cole, I.S., Chan, W.Y., Trinidad, G.S. and Paterson, D.A., 2004, Holistic model for atmospheric corrosion - Part 4- Geographic iInformation system for predicting airborne salinity. *Corros Eng Sci Techn* 39(1): 89-96.

Cole, I.S., Paterson, D.A., Ganther, W.D., Neufeld, A., Hinton, B., McAdam, G., McGeachie, M., Jeffery, R., Chotimongkol, L., Bhamornsut, C., Hue, NV. and Purwadaria, S.,.2003, Holistic model for atmospheric corrosion - Part 3 - Effect of natural and man-made landforms on deposition of marine salts in Australia and south-east Asia. *Corros Eng Sci Techn* 38(4) 267-274.

Cole, I.S., Ganther, W.G., Paterson, D.A., King, G.A., Furman, S.A. and Lau, D., 2003, Holistic model for atmospheric corrosion - Part 2 - Experimental measurement of deposition of marine salts in a number of long range studies. *Corros Eng Sci Techn* 38(4) 259-266.

Cole, I.S., Paterson, D.A. and Ganther, W.D., 2003, Holistic model for atmospheric corrosion - Part 1 Theoretical framework for production, transportation and deposition of marine salts. , *Corros Eng Sci Techn* (2) 129-134.

Cole, I.S., Furman, S.A. and Ganther, W.D., 2001, A holistic model of atmospheric corrosion. *Elec Soc S 2001*(22) 722-732.

Cole, I., Trinidad, G., Bradbury, A., McFallen, S., Chen, S.-E., MacKee, J., Gilbert, D. and Shutt, G., 2004, Final Report of Delphi study, CRC Report No 2002-020-B.

Corrosion Mapping System at www.corp.indgalv.com.au

CRC Report 2002-059-B No. 5 "Corrosion Degradation Models for Metallic Building Components".

Furman, S.A., Scholes, F.H., Hughes, A.E., Lau, D. 2005. Chromate leaching from inhibited primers II: modelling of leaching. Submitted to Progress in Organic Coatings.

Ganther, W.D., Cole, I.S., 2002. Building envelope corrosion. Proceedings of Corrosion and Prevention, Adelaide, November, 2002. Australasian Corrosion Association. Paper 067.

Gero, J.S. 1990, Design prototypes: A knowledge representation schema for design, *AI Magazine*, 11(4), 26-36.

Gero, J.S., 1998, Towards a model of designing which includes its situatedness, in H. Grabowski, S. Rude and G. Green (eds), *Universal Design Theory*, Shaker Verlag, Aachen, pp. 47-56.

Gero J.S., 1999, Constructive memory in design thinking, in G. Goldschmidt and W. Porter (eds), *Design Thinking Research Symposium: Design Representation*, MIT, Cambridge, pp. I.29-35.

Gero, J.S. and Kulinski, J., 2000, A situated approach to analogy in designing, *in* B.-K. Tang, M. Tan and Y.-C. Wong (eds), *CAADRIA2000*, CASA, Singapore, pp. 225-234.

Giarratano, J.C. and Riley, G., 1989, *Expert systems: Principles and programming*, PWS-KENT, Boston.

Howard, R.L., Zin, I.M., Scantlebury, J.D., Lyon, S.B., 1999. Inhibition of cut edge corrosion of coil-coated architectural cladding. Progress in Organic Coatings, 37, 83-90.

International Standard Organization 2000, *Buildings and Constructed Assets – Service Life Planning – Part 1: General Principles*, ISO 15686-1:2000, ISO, Geneva.

ISO 9223, "Corrosion of Metals and Alloys – Corrosivity of Atmospheres – Classification", International Organization for Standardization, Genéve, Switzerland (1992).

King, G.A., Kao, P., Norberg, P., O'Brien, D.J., 2001. Metals and coated metal products in marine environments. CSIRO Building, Construction and Engineering, Internal report, BCE Doc. 01-259.

King, G.A., Martin, K. G. and Moresby, J.F. (1982) "A detailed Corrosivity Survey of Melbourne", CSIRO, Division of Building, Construction and Engineering. ISBN 0 643 02989 3.

Liew, P.S. and Gero, J.S., 2002a, An implementation model of constructive memory for a situated design agent, *in* J.S. Gero and F. Brazier (eds), *Agents in Design 2002*, Key Centre of Design Computing and Cognition, University of Sydney, Australia, pp. 257-276.

Liew, P.S. and Gero, J.S., 2002b, A memory system for a situated design agent based on constructive memory, *in* A. Eshaq, et al. (eds), *CAADRIA2002*, Prentice Hall, New York, pp. 199-206.

Liew, P.S. and Gero, J.S., 2004, Constructive memory for situated agents, *AIEDAM (Intelligent Agents in Design)*, 18(3), 163-198.

Liew, PS and Maher, ML, 2004, Situated Case-Based reasoning as a constructive memory model of design reasoning, in HS Lee and JW Choi (eds) *The Proceedings of CAADRIA 2004*, Yonsei University Press,, Seoul, Korea, pp. 199-208.

Maher, M.L., Balachandran, M.B. and Zhang, D.M., 1995, *Case-based reasoning in design*, Lawrence Erlbaum Associates, Mahwah, N.J.

Prosek, T., Thierry, D., 2004. A model for the release of chromate from organic coatings. Progress in Organic Coatings, 49, 209-217.

Rosenman, M.A., Gero, J.S. and Oxman, R.E., 1991, What's in a case: The use of case bases, knowledge bases and databases in design, in G.N. Schmitt (ed), *CAAD Futures '91*, ETH, Zurich, pp. 263-277.

Scholes, F.H., Furman, S.A., Hughes, A.E., Nikpour, T., Wright, N., Curtis, P.R., Macrae, C.M., Intem, S., Hill, A.J., 2005. Chromate leaching from inhibited primers I: characterisation of leaching. Submitted to Progress in Organic Coatings.

Sinko, J., 2001. Challenges of chromate inhibitor pigments replacement in organic coatings. Progress in Organic Coatings, 42, 267-282.

Sirivivatnanon, V., 'Service Life Design for Environmental Loads', Keynote Address, *Proceedings of the First International Conference of Asian Concrete Federation, Vol. 1, page 1-19*, Chiang Mai, Thailand, October 28-29, 2004.

Sjöström, C. Service Life Analysis of Organic Coatings on Sheet Metal Façade Claddings. Thesis, KTH, Royal Institute of Technology, Stockholm, 1990.Suh, N.P., 1990, *The principles of design*, Oxford University Press, New York.

Wang, H., Preusel, F., Kelly, R.G., 2004. Computational modelling of inhibitor release and transport from multifunctional organic coatings. Electrochimica Acta, 49, 239-255.

Witten I.H. and Frank, E., 2000, *Data mining: Practical machine learning tools and techniques with Java implementations*, Morgan Kaufmann, San Francisco, Calif.

Zin, I.M., Howard, R.L., Badger, S.J., Scantlebury, J.D., Lyon, S.B., 1998. The mode of action of chromate inhibitor in epoxy primer on galvanised steel. Progress in Organic Coatings, 33, 203-210.

10. GLOSSARY

11. APPENDICES

Appendix I Example of Delphi Database

An example of the information stored in the Delphi database is given. This is a subset of information for over 30 building components.

Building Type	Component	Measure	Environment	Material	Maintenance	Mode (years)	SD (years)	Mean (Years)	Criteria
Commercial	Gutters	Service Life	Marine	Galvanised Steel	No	5-10	5	9	2
Commercial	Gutters	Time to First Maintenance	Marine	Galvanised Steel	Yes	<5	4	6	2
Commercial	Gutters	Aesthetic Life	Marine	Galvanised Steel	Yes	10-15	6	11	2
Commercial	Gutters	Service Life	Industrial	Galvanised Steel	Yes	10-15	9	15	2
Commercial	Gutters	Service Life	Industrial	Galvanised Steel	No	5-10	5	10	2
Commercial	Gutters	Time to First Maintenance	Industrial	Galvanised Steel	Yes	5-10	5	8	2
Commercial	Gutters	Aesthetic Life	Industrial	Galvanised Steel	Yes	5-10	6	10	2
Commercial	Gutters	Service Life	Benign	Galvanised Steel	Yes	30-50	16	32	2
Commercial	Gutters	Time to First Maintenance	Benign	Galvanised Steel	Yes	10-15	15	17	2
Commercial	Gutters	Aesthetic Life	Benign	Galvanised Steel	Yes	20-30-	13	22	2
Commercial	Gutters	Service Life	Marine	Colorbond®	No	5-10	12	18	2
Commercial	Gutters	Time to First Maintenance	Marine	Colorbond®	Yes	5-10	7	10	2
Commercial	Gutters	Service Life	Industrial	Colorbond®	Yes	15-20	14	26	2
Commercial	Gutters	Service Life	Industrial	Colorbond®	No	10-15	12	21	2
Commercial	Gutters	Time to First Maintenance	Industrial	Colorbond®	Yes	5-10	7	12	2
Commercial	Gutters	Aesthetic Life	Industrial	Colorbond®	Yes	15-20	10	17	2
Commercial	Gutters	Service Life	Benign	Colorbond®	Yes	30-50	16	36	2
Commercial	Gutters	Service Life	Benign	Colorbond®	No	30-50	16	35	2
Commercial	Gutters	Aesthetic Life	Benign	Colorbond®	Yes	30-50	14	29	2
Commercial	Gutters	Service Life	Marine	Zincalume	No	10-15	11	15	2
Commercial	Gutters	Time to First Maintenance	Marine	Zincalume	Yes	5-10	8	10	2
Commercial	Gutters	Service Life	Industrial	Zincalume	Yes	15-20	10	24	2

Appendix II Example of Maintenance Database

An example of the information stored in Maintenance Database is given.

Centre Code	CentreName	Long Deg	Lat Deg	<10 km	CaseLocation	Dist From Case	Case Long	Case Lat	Material	Service Life (years)	No of Cases
801	Aitkenvale State School	146.76	-19.29	1	VINCENT	1.0	146.77	-19.28	GAL/ZINC (UNPAINTED)	33.6	29
801	Aitkenvale State School	146.76	-19.29	1	VINCENT	1.0	146.77	-19.28	COLOURBOND	38.0	1
801	Aitkenvale State School	146.76	-19.29	1	VINCENT	1.0	146.77	-19.28	GAL/ZINC (PAINTED)	38.8	164
190	Albany Creek State School	152.97	-27.34	1	ACACIA RIDGE	4.6	153.02	-27.35	GAL/ZINC (PAINTED)	42.6	8
190	Albany Creek State School	152.97	-27.34	1	ACACIA RIDGE	4.6	153.02	-27.35	GAL/ZINC (UNPAINTED)	43.0	1
190	Albany Creek State School	152.97	-27.34	1	ACACIA RIDGE	4.6	153.02	-27.35	ALUMINIUM	52.2	29
190	Albany Creek State School	152.97	-27.34	1	ACACIA RIDGE	4.6	153.02	-27.35	COLOURBOND	45.0	1
1892	Albany Hills State School	152.97	-27.35	1	ACACIA RIDGE	4.4	153.02	-27.35	GAL/ZINC (PAINTED)	42.6	8
1892	Albany Hills State School	152.97	-27.35	1	ACACIA RIDGE	4.4	153.02	-27.35	GAL/ZINC (UNPAINTED)	43.0	1
1892	Albany Hills State School	152.97	-27.35	1	ACACIA RIDGE	4.4	153.02	-27.35	ALUMINIUM	52.2	29
1892	Albany Hills State School	152.97	-27.35	1	ACACIA RIDGE	4.4	153.02	-27.35	COLOURBOND	45.0	1
38	Albert State School	152.7	-25.54	0	NEWTOWN	0.7	152.70	-25.53	GAL/ZINC (UNPAINTED)	44.3	6
2017	Aldridge State High School	152.68	-25.51	0	MARYBOROUGH	3.0	152.70	-25.53	GAL/ZINC (PAINTED)	53.0	8
2206	Allenstown Special Education Unit	150.5	-23.39	0	NORTH ROCKHAMPTON	1.9	150.52	-23.38	GAL/ZINC (UNPAINTED)	33.8	46
2206	Allenstown Special Education Unit	150.5	-23.39	0	NORTH ROCKHAMPTON	1.9	150.52	-23.38	COLOURBOND	27.0	1

2206	Allenstown Special Education Unit	150.5	-23.39	0	NORTH ROCKHAMPTON	1.9	150.52	-23.38	GAL/ZINC (PAINTED)	41.8	13
155	Allenstown State School	150.5	-23.39	0	NORTH ROCKHAMPTON	1.9	150.52	-23.38	GAL/ZINC (UNPAINTED)	33.8	46
155	Allenstown State School	150.5	-23.39	0	NORTH ROCKHAMPTON	1.9	150.52	-23.38	COLOURBOND	27.0	1
155	Allenstown State School	150.5	-23.39	0	NORTH ROCKHAMPTON	1.9	150.52	-23.38	GAL/ZINC (PAINTED)	41.8	13

Appendix III Computation of distance between two points on the Earth's surface

The distance, *D*, between two points on the surface on the earth is computed by the following formula:

$$D = R \times \cos^{-1}\left\{\left[\sin\left(\frac{latitude_1}{57.2958}\right) \times \sin\left(\frac{latitude_2}{57.2958}\right)\right] + \left[\cos\left(\frac{latitude_1}{57.2958}\right) \times \cos\left(\frac{latitude_2}{57.2958}\right) \times \cos\left(\frac{longitude_2}{57.2958} - \frac{longitude_1}{57.2958}\right)\right]\right\}$$

where:

the location of the first point is given by (*longitude*$_1$, *latitude*$_1$);
the location of the second point is given by (*longitude*$_2$, *latitude*$_2$);
and longitudes and latitudes are measure in decimal degrees;
R is the radius of the earth taken as 6378.7 km.

To convert latitude or longitude from decimal degrees to radians, the latitude and longitude values are divided by $\frac{180}{\pi} \approx 57.2956$ (taking π to be 3.1416).

Appendix IV Java Classes for CBR code

Package **Class** Tree Deprecated Index Help

PREV CLASS NEXT CLASS FRAMES NO FRAMES All Classes

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

Class ComponentLifeCase

```
java.lang.Object
  ComponentLifeCase
```

All Implemented Interfaces:

ComponentLifeSubCase, java.io.Serializable

---

public class **ComponentLifeCase**

extends java.lang.Object

implements java.io.Serializable, ComponentLifeSubCase

See Also:

Serialized Form

Constructor Summary

ComponentLifeCase() Creates a new instance of ComponentLifeCase
ComponentLifeCase(java.lang.String name, ComponentLifeTableInput tableInput, ComponentLifeUserInput initialInputData, java.util.Date timeStamp) Creates a new instance of ComponentLifeCase
ComponentLifeCase(java.lang.String id, java.util.Date timeStamp, ComponentLifeUserInput initialInputData, java.util.Vector alternatives, ComponentLifeUserInput finalInputData, java.lang.String module, double value, double similarityIndex, ComponentLifeTableInput similarityTable) Creates a new instance of ComponentLifeCase

Method Summary

void	**displayContents**(int prefixNumOfTabs) Displays the contents of the case.
java.util.Vector	**getAlternatives**() Getter for property alternatives.

double	**getDistance**() Getter for property distance.
ComponentLifeUserInput	**getFinalInputData**() Getter for property finalInputData.
java.lang.String	**getId**() Getter for property id.
ComponentLifeUserInput	**getInitialInputData**() Getter for property initialInputData.
java.lang.String	**getModule**() Getter for property module.
double	**getSimilarityIndex**() Getter for property similarityIndex.
double	**getSimilarityIndex**(ComponentLifeUserInput input) Computes the similarity index between this case and the situation defined by the input data.
double	**getValue**() Getter for property value.
void	**setAlternatives**(java.util.Vector alternatives) Setter for property alternatives.
void	**setDistance**(double distance) Setter for property distance.
void	**setFinalInputData**(ComponentLifeUserInput finalInputData) Setter for property finalInputData.
void	**setId**(java.lang.String id) Setter for property id.
void	**setInitialInputData**(ComponentLifeUserInput initialInputData) Setter for property initialInputData.
void	**setModule**(java.lang.String module) Setter for property module.
void	**setSimilarityIndex**(double similarityIndex) Setter for property similarityIndex.
void	**setValue**(double value) Setter for property value.
java.lang.String	**toString**() Returns a String representation of the case.

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

ComponentLifeCase

```
public ComponentLifeCase()
```

Creates a new instance of ComponentLifeCase

ComponentLifeCase

```
public ComponentLifeCase(java.lang.String id,
                         java.util.Date timeStamp,
                         ComponentLifeUserInput initialInputData,
                         java.util.Vector alternatives,
                         ComponentLifeUserInput finalInputData,
                         java.lang.String module,
                         double value,
                         double similarityIndex,
                         ComponentLifeTableInput similarityTable)
```

Creates a new instance of ComponentLifeCase

ComponentLifeCase

```
public ComponentLifeCase(java.lang.String name,
                         ComponentLifeTableInput tableInput,
                         ComponentLifeUserInput initialInputData,
                         java.util.Date timeStamp)
```

Creates a new instance of ComponentLifeCase

Method Detail

getSimilarityIndex

```
public double getSimilarityIndex(ComponentLifeUserInput input)
```

Computes the similarity index between this case and the situation defined by the input data.

---

toString

```
public java.lang.String toString()
```

Returns a String representation of the case.

Specified by:

`toString` in interface `ComponentLifeSubCase`

---

getInitialInputData

```
public ComponentLifeUserInput getInitialInputData()
```

Getter for property initialInputData.

Returns:

Value of property initialInputData.

---

setInitialInputData

`public void` **`setInitialInputData`**`(ComponentLifeUserInput initialInputData)`

Setter for property initialInputData.

Parameters:

`initialInputData` - New value of property initialInputData.

---

getModule

`public java.lang.String` **`getModule`**`()`

Getter for property module.

Returns:

Value of property module.

---

setModule

`public void` **`setModule`**`(java.lang.String module)`

Setter for property module.

Parameters:

`module` - New value of property module.

---

getId

`public java.lang.String` **`getId`**`()`

Getter for property id.

Returns:

Value of property id.

---

setId

`public void` **`setId`**`(java.lang.String id)`

Setter for property id.

Parameters:

`id` - New value of property id.

---

getValue

`public double` **`getValue`**`()`

Getter for property value.

Returns:

Value of property value.

---

setValue

```
public void setValue(double value)
```

Setter for property value.

Parameters:

`value` - New value of property value.

---

getFinalInputData

```
public ComponentLifeUserInput getFinalInputData()
```

Getter for property finalInputData.

Returns:

Value of property finalInputData.

---

setFinalInputData

```
public void setFinalInputData(ComponentLifeUserInput finalInputData)
```

Setter for property finalInputData.

Parameters:

`finalInputData` - New value of property finalInputData.

---

getAlternatives

```
public java.util.Vector getAlternatives()
```

Getter for property alternatives.

Returns:

Value of property alternatives.

---

setAlternatives

```
public void setAlternatives(java.util.Vector alternatives)
```

Setter for property alternatives.

Parameters:

`alternatives` - New value of property alternatives.

---

getSimilarityIndex

```
public double getSimilarityIndex()
```

Getter for property similarityIndex.

Returns:

Value of property similarityIndex.

setSimilarityIndex

`public void` **`setSimilarityIndex`**`(double similarityIndex)`

Setter for property similarityIndex.

Parameters:

`similarityIndex` - New value of property similarityIndex.

getDistance

`public double` **`getDistance`**`()`

Getter for property distance.

Returns:

Value of property distance.

setDistance

`public void` **`setDistance`**`(double distance)`

Setter for property distance.

Parameters:

`distance` - New value of property distance.

displayContents

`public void` **`displayContents`**`(int prefixNumOfTabs)`

Displays the contents of the case.

Class ComponentLifeCaseBase

```
java.lang.Object
  ComponentLifeCaseBase
```

All Implemented Interfaces:

ComponentLifeDataSource, java.io.Serializable

public class **ComponentLifeCaseBase**

extends java.lang.Object

implements java.io.Serializable, ComponentLifeDataSource

This is a wrapper class for the Casebase.

See Also:

Serialized Form

Constructor Summary

ComponentLifeCaseBase()	Creates a new instance of ComponentLifeCaseBase

Method Summary

void	**addCase**(ComponentLifeCase icase) Adds a new case to the casebase.
void	**displayContents**(int prefixNumOfTabs) Displays the contents of the casebase.
java.util.Vector	**getAlternatives**(ComponentLifeUserInput input, java.util.Date timeStamp, java.lang.String id) Get alternatives from the casebase for interpretation and construction.
java.util.Vector	**getAlternativesAsStrings**(ComponentLifeUserInput userInput) Returns the String representations of all alternative cases as a collection.
ComponentLifeCase	**getCase**(int index) Gets a case from the casebase according to an index.

ComponentLifeCase	**getCase**(java.lang.String caseID) Gets a case based on its identifier.
java.util.Vector	**getSimilarCases**(ComponentLifeUserInput userInput)
ComponentLifeTableInput	**getSimilarityTable**() Getter for property similarityTable.
int	**getSize**()
void	**setSimilarityTable**(ComponentLifeTableInput similarityTable) Setter for property similarityTable.
java.lang.String	**toString**()

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

ComponentLifeCaseBase

```
public ComponentLifeCaseBase()
```
Creates a new instance of ComponentLifeCaseBase

Method Detail

addCase

```
public void addCase(ComponentLifeCase icase)
```
Adds a new case to the casebase.

---

getCase

```
public ComponentLifeCase getCase(int index)
```
Gets a case from the casebase according to an index.

---

getSize

```
public int getSize()
```

---

toString

```
public java.lang.String toString()
```

---

getSimilarityTable

```
public ComponentLifeTableInput getSimilarityTable()
```

Getter for property similarityTable.

Returns:

Value of property similarityTable.

---

setSimilarityTable

```
public void setSimilarityTable(ComponentLifeTableInput similarityTable)
```

Setter for property similarityTable.

Parameters:

`similarityTable` - New value of property similarityTable.

---

getSimilarCases

```
public java.util.Vector getSimilarCases(ComponentLifeUserInput userInput)
```

---

getAlternatives

```
public java.util.Vector getAlternatives(ComponentLifeUserInput input,
                                        java.util.Date timeStamp,
                                        java.lang.String id)
```

Get alternatives from the casebase for interpretation and construction. The current way to get alternatives is to compute the similarity index of all cases within the casebase and select those with similarity value greater than the thresold.

Specified by:

`getAlternatives` in interface `ComponentLifeDataSource`

---

getAlternativesAsStrings

```
public java.util.Vector
getAlternativesAsStrings(ComponentLifeUserInput userInput)
```

Returns the String representations of all alternative cases as a collection.

---

getCase

```
public ComponentLifeCase getCase(java.lang.String caseID)
```

Gets a case based on its identifier.

---

displayContents

```
public void displayContents(int prefixNumOfTabs)
```

Displays the contents of the casebase.

Class ComponentLifeConsoleDisplay

```
java.lang.Object
  ComponentLifeConsoleDisplay
```

public class **ComponentLifeConsoleDisplay**

extends java.lang.Object

A placeholder to display results for throws situatedCBR systems in the screen console until a graphical user interface is developed.

Constructor Summary

ComponentLifeConsoleDisplay()	Creates a new instance of ComponentLifeConsoleDisplay

Method Summary

void	**displayComponentLifeClass**(java.lang.String str, int prefixNumOfTabs) Display the String representations of objects in the system after prefixing each new line with a specified number of "tab" characters.
java.lang.String	**getConsoleDisplayString**(java.lang.String str, int prefixNumOfTabs) Prefix a specified number of "tab" characters to the String representations of objects in the system.

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

ComponentLifeConsoleDisplay

```
public ComponentLifeConsoleDisplay()
```

Creates a new instance of ComponentLifeConsoleDisplay

Method Detail

displayComponentLifeClass

```
public void displayComponentLifeClass(java.lang.String str,
                                      int prefixNumOfTabs)
```

Display the String representations of objects in the system after prefixing each new line with a specified number of "tab" characters.

---

getConsoleDisplayString

```
public java.lang.String getConsoleDisplayString(java.lang.String str,
                                                 int prefixNumOfTabs)
```

Prefix a specified number of "tab" characters to the String representations of objects in the system.

Class ComponentLifeConstructor

```
java.lang.Object
  ComponentLifeConstructor
```

public class **ComponentLifeConstructor**

extends java.lang.Object

Performs the constructive functions as dictated by the situated CBR model. A new case is created by this construction.

Constructor Summary

ComponentLifeConstructor()	Creates a new instance of ComponentLifeConstructor

Method Summary

void	**activateInferenceEngine**(ComponentLifeUserInput userInput, java.util.Vector alternatives, ComponentLifeCase theCase) Provides an entry point to an external inference engine to attached to the situated CBR system for constructing a new case according to a set of alternatives obtained from the casebase, holistic model, delphi database and field database based on domain heuristics.
void	**constructCase**(java.util.Vector dataSources, ComponentLifeUserInput userInput, java.lang.String idPrefix, java.util.Date timeStamp, ComponentLifeCase theCase) Construct a new case through the use of domain heuristics operating on the set of alternatives obtained from the casebase, holistic model, delphi database and field database.
void	**displayAlternatives**(java.util.Vector alternatives) Display the alternatives from the casebase, holistic model, delphi database and field database (represented as a Vector of Vectors) as a String oblect with one "tab" characters as prefix.
java.util.Vector	**getAlternatives**(java.util.Vector dataSources, ComponentLifeUserInput userInput, java.lang.String idPrefix, java.util.Date timeStamp)

	Returns a Vector of Vectors object that represents the alternatives from the casebase, holistic model, delphi database and field database based on the user input parameter values.

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

ComponentLifeConstructor

```
public ComponentLifeConstructor()
```

Creates a new instance of ComponentLifeConstructor

Method Detail

constructCase

```
public void constructCase(java.util.Vector dataSources,
                          ComponentLifeUserInput userInput,
                          java.lang.String idPrefix,
                          java.util.Date timeStamp,
                          ComponentLifeCase theCase)
```

Construct a new case through the use of domain heuristics operating on the set of alternatives obtained from the casebase, holistic model, delphi database and field database.

---

getAlternatives

```
public java.util.Vector getAlternatives(java.util.Vector dataSources,
                                        ComponentLifeUserInput userInput,
                                        java.lang.String idPrefix,
                                        java.util.Date timeStamp)
```

Returns a Vector of Vectors object that represents the alternatives from the casebase, holistic model, delphi database and field database based on the user input parameter values.

Each sub-Vector object represents alternatives from either the casebase, holistic model, delphi database or field database.

---

displayAlternatives

`public void` **`displayAlternatives`**`(java.util.Vector alternatives)`

Display the alternatives from the casebase, holistic model, delphi database and field database (represented as a Vector of Vectors) as a String oblect with one "tab" characters as prefix.

---

activateInferenceEngine

```
public void activateInferenceEngine(ComponentLifeUserInput userInput,
                                    java.util.Vector alternatives,
                                    ComponentLifeCase theCase)
```

Provides an entry point to an external inference engine to attached to the situated CBR system for constructing a new case according to a set of alternatives obtained from the casebase, holistic model, delphi database and field database based on domain heuristics.

No inference engine is used in the current version of the system. The alternatives are displayed and the method to compute value is set to a string "PLACEHOLDER". The computed value is set to 100.. The vector of vector representing the alternatives is not used.

Interface ComponentLifeDataSource

All Known Implementing Classes:

ComponentLifeCaseBase, ComponentLifeDelphiDatabase, ComponentLifeFieldDatabase, ComponentLifeHolisticModel

public interface **ComponentLifeDataSource**

An interface that all data sources must conform to. Currently these sources are: ComponentLifeCaseBase, ComponentLifeHolisticModel, ComponentLifeDelphiDatabase, ComponentLifeFieldDatabase.

Method Summary

java.util.Vector	**getAlternatives**(ComponentLifeUserInput input, java.util.Date timeStamp, java.lang.String id)

Method Detail

getAlternatives

```
public java.util.Vector getAlternatives(ComponentLifeUserInput input,
                                        java.util.Date timeStamp,
                                        java.lang.String id)
```

Class ComponentLifeDelphiDatabase

```
java.lang.Object
  ComponentLifeDelphiDatabase
```

All Implemented Interfaces:

ComponentLifeDataSource

public class **ComponentLifeDelphiDatabase**

extends java.lang.Object

implements ComponentLifeDataSource

This is a wrapper class for the Delphi Database.

Constructor Summary

ComponentLifeDelphiDatabase()	Creates a new instance of ComponentLifeDelphiDatabase

Method Summary

java.util.Vector	**getAlternatives**(ComponentLifeUserInput input, java.util.Date timeStamp, java.lang.String id) Get alternatives from the Holistic model for interpretation and construction.
double	**getLifeSurvey**(java.lang.String compType, java.lang.String material, ComponentLifeGeoLocation geoLoc) Gets predicted component life value.

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

ComponentLifeDelphiDatabase

```
public ComponentLifeDelphiDatabase()
```

Creates a new instance of ComponentLifeDelphiDatabase

Method Detail

getLifeSurvey

```
public double getLifeSurvey(java.lang.String compType,
                            java.lang.String material,
                            ComponentLifeGeoLocation geoLoc)
```

Gets predicted component life value.

---

getAlternatives

```
public java.util.Vector getAlternatives(ComponentLifeUserInput input,
                                        java.util.Date timeStamp,
                                        java.lang.String id)
```

Get alternatives from the Holistic model for interpretation and construction. The current way to get alternatives is based on using different materials and these alternatives are returned as objects of ComponentLifeHolisticSubCase. The original input is also used to generate an alternative.

Specified by:

getAlternatives in interface ComponentLifeDataSource

Class ComponentLifeDelphiSubCase

```
java.lang.Object
  ComponentLifeDelphiSubCase
```

All Implemented Interfaces:

ComponentLifeSubCase, java.io.Serializable

public class **ComponentLifeDelphiSubCase**

extends java.lang.Object

implements java.io.Serializable, ComponentLifeSubCase

This class represents an alternative from the Delphi Database that the system can used during interpretation or construction.

See Also:

Serialized Form

Constructor Summary

ComponentLifeDelphiSubCase(java.lang.String id, java.util.Date timeStamp, ComponentLifeUserInput userInput) Creates a new instance of ComponentLifeDelphiSubCase

Method Summary

java.lang.String	**toString**() Return a String representation of the object

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

ComponentLifeDelphiSubCase

```
public ComponentLifeDelphiSubCase(java.lang.String id,
                                  java.util.Date timeStamp,
                                  ComponentLifeUserInput userInput)
```

Creates a new instance of ComponentLifeDelphiSubCase

Method Detail

toString

```
public java.lang.String toString()
```

Return a String representation of the object

Specified by:

`toString` in interface `ComponentLifeSubCase`

Class ComponentLifeFieldDatabase

```
java.lang.Object
  ComponentLifeFieldDatabase
```

All Implemented Interfaces:

ComponentLifeDataSource

public class **ComponentLifeFieldDatabase**

extends java.lang.Object

implements ComponentLifeDataSource

This is a wrapper class for the Field (Maintenance) Database.

Constructor Summary

ComponentLifeFieldDatabase() Creates a new instance of ComponentLifeFieldDatabase

Method Summary

java.util.Vector	**getAlternatives**(ComponentLifeUserInput input, java.util.Date timeStamp, java.lang.String id) Get alternatives from the Holistic model for interpretation and construction.
double	**getLifeData**(java.lang.String compType, java.lang.String material, ComponentLifeGeoLocation geoLoc) Gets predicted component life value.

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

ComponentLifeFieldDatabase

```
public ComponentLifeFieldDatabase()
```

Creates a new instance of ComponentLifeFieldDatabase

Method Detail

getLifeData

```
public double getLifeData(java.lang.String compType,
                          java.lang.String material,
                          ComponentLifeGeoLocation geoLoc)
```

Gets predicted component life value.

---

getAlternatives

```
public java.util.Vector getAlternatives(ComponentLifeUserInput input,
                                        java.util.Date timeStamp,
                                        java.lang.String id)
```

Get alternatives from the Holistic model for interpretation and construction. The current way to get alternatives is based on using different materials and these alternatives are returned as objects of ComponentLifeHolisticSubCase. The original input is also used to generate an alternative.

Specified by:

`getAlternatives` in interface `ComponentLifeDataSource`

Class ComponentLifeFieldSubCase

java.lang.Object
  ComponentLifeFieldSubCase

All Implemented Interfaces:

ComponentLifeSubCase, java.io.Serializable

---

public class **ComponentLifeFieldSubCase**

extends java.lang.Object

implements java.io.Serializable, ComponentLifeSubCase

This class represents an alternative from the Field Database that the system can used during interpretation or construction.

See Also:

Serialized Form

Constructor Summary

ComponentLifeFieldSubCase(java.lang.String id, java.util.Date timeStamp, ComponentLifeUserInput userInput) Creates a new instance of ComponentLifeFieldSubCase

Method Summary

java.lang.String	**toString**() Return a String representation of the object

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

ComponentLifeFieldSubCase

```
public ComponentLifeFieldSubCase(java.lang.String id,
                                 java.util.Date timeStamp,
                                 ComponentLifeUserInput userInput)
```

Creates a new instance of ComponentLifeFieldSubCase

Method Detail

toString

```
public java.lang.String toString()
```

Return a String representation of the object

Specified by:

`toString` in interface `ComponentLifeSubCase`

Class ComponentLifeGeoLocation

```
java.lang.Object
  ComponentLifeGeoLocation
```

All Implemented Interfaces:

java.io.Serializable

public class **ComponentLifeGeoLocation**

extends java.lang.Object

implements java.io.Serializable

This is a representation of a geographic location.

See Also:

Serialized Form

Constructor Summary

ComponentLifeGeoLocation()	Creates a new instance of ComponentLifeGeoLocation
ComponentLifeGeoLocation(double locX, double locY)	Creates a new instance of ComponentLifeGeoLocation with two doubles

Method Summary

double	**getDistance**(ComponentLifeGeoLocation geoLocation) Computes the distance between two geoggraphic locations
double	**getLocationX**() Getter for property locationX.
double	**getLocationY**() Getter for property locationY.
static void	**main**(java.lang.String[] args) Unit testing function.
void	**setLocationX**(double locationX) Setter for property locationX.

void	**setLocationY**(double locationY) Setter for property locationY.
java.lang.String	**toString**() Returns a String representation of the object.

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

ComponentLifeGeoLocation

```
public ComponentLifeGeoLocation()
```

Creates a new instance of ComponentLifeGeoLocation

ComponentLifeGeoLocation

```
public ComponentLifeGeoLocation(double locX,
                                double locY)
```

Creates a new instance of ComponentLifeGeoLocation with two doubles

Method Detail

getLocationX

```
public double getLocationX()
```

Getter for property locationX.

Returns:

Value of property locationX.

---

setLocationX

```
public void setLocationX(double locationX)
```

Setter for property locationX.

Parameters:

`locationX` - New value of property locationX.

---

getLocationY

```
public double getLocationY()
```

Getter for property locationY.

Returns:

Value of property locationY.

---

setLocationY

```
public void setLocationY(double locationY)
```

Setter for property locationY.

Parameters:

`locationY` - New value of property locationY.

---

getDistance

```
public double getDistance(ComponentLifeGeoLocation geoLocation)
```

Computes the distance between two geoggraphic locations

---

toString

```
public java.lang.String toString()
```

Returns a String representation of the object.

---

main

```
public static void main(java.lang.String[] args)
```

Unit testing function.

Class ComponentLifeHolisticModel

```
java.lang.Object
  ComponentLifeHolisticModel
```

All Implemented Interfaces:

ComponentLifeDataSource

public class **ComponentLifeHolisticModel**

extends java.lang.Object

implements ComponentLifeDataSource

This is a wrapper class for the Holistic Model.

Constructor Summary

ComponentLifeHolisticModel()	Creates a new instance of ComponentLifeHolisticModel

Method Summary

java.util.Vector	**getAlternatives**(ComponentLifeUserInput input, java.util.Date timeStamp, java.lang.String id) Get alternatives from the Holistic model for interpretation and construction.
double	**getLifeModel**(java.lang.String compType, java.lang.String material, ComponentLifeGeoLocation geoLoc) Gets predicted component life value.
double	**getSalt**(ComponentLifeGeoLocation geoLoc) Gets Salinity value based on location data.
double	**getToW**(ComponentLifeGeoLocation geoLoc) Gets Time-to-Wetness value based on location data.
static void	**main**(java.lang.String[] args) Unit testing function.

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

ComponentLifeHolisticModel

```
public ComponentLifeHolisticModel()
```

Creates a new instance of ComponentLifeHolisticModel

Method Detail

getToW

```
public double getToW(ComponentLifeGeoLocation geoLoc)
```

Gets Time-to-Wetness value based on location data.

---

getSalt

```
public double getSalt(ComponentLifeGeoLocation geoLoc)
```

Gets Salinity value based on location data.

---

getLifeModel

```
public double getLifeModel(java.lang.String compType,
                           java.lang.String material,
                           ComponentLifeGeoLocation geoLoc)
```

Gets predicted component life value.

---

getAlternatives

```
public java.util.Vector getAlternatives(ComponentLifeUserInput input,
                                        java.util.Date timeStamp,
                                        java.lang.String id)
```

Get alternatives from the Holistic model for interpretation and construction. The current way to get alternatives is based on using different materials and these alternatives are returned as objects of ComponentLifeHolisticSubCase. The original input is also used to generate an alternative.

Specified by:

getAlternatives in interface ComponentLifeDataSource

---

main

```
public static void main(java.lang.String[] args)
```

Unit testing function.

Class ComponentLifeHolisticSubCase

java.lang.Object
  ComponentLifeHolisticSubCase

All Implemented Interfaces:

ComponentLifeSubCase, java.io.Serializable

public class **ComponentLifeHolisticSubCase**

extends java.lang.Object

implements java.io.Serializable, ComponentLifeSubCase

This class represents an alternative from the Holistic Model that the system can used during interpretation or construction.

See Also:

Serialized Form

Constructor Summary

ComponentLifeHolisticSubCase(java.lang.String id, java.util.Date timeStamp, ComponentLifeUserInput userInput) Creates a new instance of ComponentLifeHolisticSubCase

Method Summary

java.lang.String	**toString**() Return a String representation of the object

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

ComponentLifeHolisticSubCase

```
public ComponentLifeHolisticSubCase(java.lang.String id,
                                    java.util.Date timeStamp,
                                    ComponentLifeUserInput userInput)
```

Creates a new instance of ComponentLifeHolisticSubCase

Method Detail

toString

```
public java.lang.String toString()
```

Return a String representation of the object

Specified by:

`toString` in interface `ComponentLifeSubCase`

Class ComponentLifeInterpreter

```
java.lang.Object
  ComponentLifeInterpreter
```

public class **ComponentLifeInterpreter**

extends java.lang.Object

Performs the interpretive functions as dictated by the situated CBR model. User input paramaters values are finalized by this class during interpretation.

Constructor Summary

ComponentLifeInterpreter()	Creates a new instance of ComponentLifeInterpreter

Method Summary

ComponentLifeUserInput	**activateInferenceEngine**(ComponentLifeUserInput userInput, java.util.Vector alternatives) Provides an entry point to an external inference engine to attached to the situated CBR system for interpreting the current situation according to a set of alternatives obtained from the casebase, holistic model, delphi database and field database based on domain heuristics.
void	**displayAlternatives**(java.util.Vector alternatives, int tabSpace) Display the alternatives from the casebase, holistic model, delphi database and field database (represented as a Vector of Vectors) as a String oblect with a specific number of "tab" characters as prefix.
java.util.Vector	**getAlternatives**(java.util.Vector dataSources, ComponentLifeUserInput userInput, java.lang.String idPrefix, java.util.Date timeStamp) Returns a Vector of Vectors object that represents the alternatives from the casebase, holistic model, delphi database and field database based on the user input parameter values.
java.lang.String	**getAlternativesStrings**(java.util.Vector alternatives) Return the alternatives from the casebase, holistic model,

	delphi database and field database, represented as a Vector of Vectors object, as a String object.
ComponentLifeUserInput	**getFinalUserInput**(java.util.Vector dataSources, ComponentLifeUserInput userInput, java.lang.String idPrefix, java.util.Date timeStamp) Finalizes the input parameter values as a new ComponentLifeUserInput object through the use of domain heuristics operating on the set of alternatives obtained from the casebase, holistic model, delphi database and field database.

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

ComponentLifeInterpreter

public **ComponentLifeInterpreter**()

Creates a new instance of ComponentLifeInterpreter

Method Detail

getFinalUserInput

public ComponentLifeUserInput
getFinalUserInput(java.util.Vector dataSources,

ComponentLifeUserInput userInput,
java.lang.String idPrefix,
java.util.Date timeStamp)

Finalizes the input parameter values as a new ComponentLifeUserInput object through the use of domain heuristics operating on the set of alternatives obtained from the casebase, holistic model, delphi database and field database.

---

getAlternatives

public java.util.Vector **getAlternatives**(java.util.Vector dataSources,
ComponentLifeUserInput userInput,
java.lang.String idPrefix,
java.util.Date timeStamp)

Returns a Vector of Vectors object that represents the alternatives from the casebase, holistic model, delphi database and field database based on the user input parameter values.

Each sub-Vector object represents alternatives from either the casebase, holistic model, delphi database or field database.

---

activateInferenceEngine

```
public ComponentLifeUserInput
activateInferenceEngine(ComponentLifeUserInput userInput,

java.util.Vector alternatives)
```

Provides an entry point to an external inference engine to attached to the situated CBR system for interpreting the current situation according to a set of alternatives obtained from the casebase, holistic model, delphi database and field database based on domain heuristics.

No inference engine is used in the current version of the system. The alternatives are displayed and the input data is duplicated. The vector of vector representing the alternatives is not used.

---

displayAlternatives

```
public void displayAlternatives(java.util.Vector alternatives,
                                int tabSpace)
```

Display the alternatives from the casebase, holistic model, delphi database and field database (represented as a Vector of Vectors) as a String oblect with a specific number of "tab" characters as prefix.

---

getAlternativesStrings

```
public java.lang.String
getAlternativesStrings(java.util.Vector alternatives)
```

Return the alternatives from the casebase, holistic model, delphi database and field database, represented as a Vector of Vectors object, as a String object.

Class ComponentLifeSituatedCBR

```
java.lang.Object
  ComponentLifeSituatedCBR
```

public class **ComponentLifeSituatedCBR**

extends java.lang.Object

This class provides the entry point to the situated CBR system via its main() function. The following procedures are followed for a typical run of the system:

1) Initialization:

a) casebase file and data file for computing similarity index is located

b) interface to data file for computing similarity index is created

c) casebase is loaded

d) holistic model is loaded

e) delphi database is loaded

f) field database is loaded

g) all casebase, holistic model and databases are collected into a Vector object as data sources

h) new case is created

2) Interpretation:

a) interpreter is created

b) user input is finalized

3) Construction:

a) constructor is created

b) all parameter valuess of the new case is computed

4) Closure

a) new case is saved

b) casebase is closed

Constructor Summary

ComponentLifeSituatedCBR() Creates a new instance of ComponentLifeSituatedCBR

Method Summary

static void	**main**(java.lang.String[] args) Entry point to functions for system testing.
void	**testScenario0**() Testing codes for Test Scenario 0
void	**testScenario1**() Testing codes for Test Scenario 1
void	**testScenario2**() Testing codes for Test Scenario 2
void	**testScenario3**() Testing codes for Test Scenario 3
void	**testScenario4**() Testing codes for Test Scenario 4
void	**testScenario5**() Testing codes for Test Scenario 5
void	**testScenario6**() Testing codes for Test Scenario 6
void	**testScenario7**() Testing codes for Test Scenario 7
void	**testScenario8**() Testing codes for Test Scenario 8

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

ComponentLifeSituatedCBR

```
public ComponentLifeSituatedCBR()
```
Creates a new instance of ComponentLifeSituatedCBR

Method Detail

testScenario0

```
public void testScenario0()
```
Testing codes for Test Scenario 0

---

testScenario1

```
public void testScenario1()
```
Testing codes for Test Scenario 1

---

testScenario2

```
public void testScenario2()
```
Testing codes for Test Scenario 2

---

testScenario3

```
public void testScenario3()
```
Testing codes for Test Scenario 3

---

testScenario4

```
public void testScenario4()
```
Testing codes for Test Scenario 4

---

testScenario5

```
public void testScenario5()
```
Testing codes for Test Scenario 5

---

testScenario6

```
public void testScenario6()
```
Testing codes for Test Scenario 6

---

testScenario7

`public void` **`testScenario7`**`()`

Testing codes for Test Scenario 7

---

testScenario8

`public void` **`testScenario8`**`()`

Testing codes for Test Scenario 8

---

main

`public static void` **`main`**`(java.lang.String[] args)`

Entry point to functions for system testing. Comment off different function calls to effect different test scenarios.

Parameters:

`args` - the command line arguments

Interface ComponentLifeSubCase

All Known Implementing Classes:

ComponentLifeCase, ComponentLifeDelphiSubCase, ComponentLifeFieldSubCase, ComponentLifeHolisticSubCase

public interface **ComponentLifeSubCase**

An interface that all subcase objects must conform to. Currently these subcases are: ComponentLifeCase, ComponentLifeHolisticSubCase, ComponentLifeDelphiSubCase, ComponentLifeFieldSubCase.

Method Summary

java.lang.String	**toString**()

Method Detail

toString

```
public java.lang.String toString()
```

Class ComponentLifeTableInput

```
java.lang.Object
  ComponentLifeTableInput
```

All Implemented Interfaces:

java.io.Serializable

public class **ComponentLifeTableInput**

extends java.lang.Object

implements java.io.Serializable

This class provides an interface to the codes that read in the data required for computing the similarity index between the situations defined by a case and the user input. Currently the data resides within the file: userInputCFG.txt.

Expansion Possibilities:

1) The tabulated within the data file (userInputCFG.txt) can be changed to provide better coorelations between different situations when computing the similarity based on: a) maintanence factor; b)cleaning factor with condition gunk can collect or cleaning factor with gunk cannot collect; c)location-in-building factor; and d) geographic-location factor with condition as marine application.

2) The whole notion of using the data file can be replaced by another approach. To isolate changes to existing codes, the functionalities as defined by the interface to this class (public methods) must be conformed to by the new implementation.

See Also:

Serialized Form

Nested Class Summary

class	**ComponentLifeTableInput.Parameter**

Constructor Summary

ComponentLifeTableInput()

Creates a new instance of ComponentLifeTableInput
ComponentLifeTableInput(java.lang.String fileName) Creates a new instance of ComponentLifeTableInput based on the contents of a data file.

Method Summary

java.lang.String	**getColumn**(java.lang.String str, int col
int	**getMaxInt**(java.lang.String str) Helper function to get the maximum value from a reprsentation of a range.
ComponentLifeTableInput.Parameter	**getParameter**() Getter for property parameter.
double	**getParameterSimilarityIndex**(java.lang.String name, int oldCaseValue, int newCas java.lang.String condition)
double	**getParameterSimilarityIndex**(java.lang.String name, java.lang.String oldCaseValue, java.lang.String newCaseValue, java.lang.String condition)
java.util.Vector	**getTableContents**() Getter for property tableContents.
java.lang.String	**getVariableName**(java.lang.String name, java.lang.String condition, int intInpu
static void	**main**(java.lang.String[] args) Unit testing function.
void	**setParameter**(ComponentLifeTableInput.Parameter par Setter for property parameter.
void	**setTableContents**(java.util.Vector tableContents) Setter for property tableContents.

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

ComponentLifeTableInput

`public` **`ComponentLifeTableInput`**`()`

Creates a new instance of ComponentLifeTableInput

ComponentLifeTableInput

`public` **`ComponentLifeTableInput`**`(java.lang.String fileName)`

Creates a new instance of ComponentLifeTableInput based on the contents of a data file.

Method Detail

getParameter

`public` `ComponentLifeTableInput.Parameter` **`getParameter`**`()`

Getter for property parameter.

Returns:

Value of property parameter.

---

setParameter

`public void` **`setParameter`**`(ComponentLifeTableInput.Parameter parameter)`

Setter for property parameter.

Parameters:

`parameter` - New value of property parameter.

---

getTableContents

`public java.util.Vector` **`getTableContents`**`()`

Getter for property tableContents.

Returns:

Value of property tableContents.

---

setTableContents

`public void` **`setTableContents`**`(java.util.Vector tableContents)`

Setter for property tableContents.

Parameters:

`tableContents` - New value of property tableContents.

---

getParameterSimilarityIndex

```
public double getParameterSimilarityIndex(java.lang.String name,
                                           java.lang.String oldCaseValue,
                                           java.lang.String newCaseValue,
                                           java.lang.String condition)
```

---

getColumn

```
public java.lang.String getColumn(java.lang.String str,
                                  int columnNum)
```

---

getParameterSimilarityIndex

```
public double getParameterSimilarityIndex(java.lang.String name,
                                           int oldCaseValue,
                                           int newCaseValue,
                                           java.lang.String condition)
```

---

getVariableName

```
public java.lang.String getVariableName(java.lang.String name,
                                        java.lang.String condition,
                                        int intInputValue)
```

---

getMaxInt

```
public int getMaxInt(java.lang.String str)
```

Helper function to get the maximum value from a String reprsentation of a range. For example: the function will return an integer of value 2 when str equals to ""

---

main

```
public static void main(java.lang.String[] args)
```

Unit testing function.

Class ComponentLifeTableInput.Parameter

```
java.lang.Object
  ComponentLifeTableInput.Parameter
```

All Implemented Interfaces:

java.io.Serializable

Enclosing class:

ComponentLifeTableInput

public class **ComponentLifeTableInput.Parameter**

extends java.lang.Object

implements java.io.Serializable

See Also:

Serialized Form

Constructor Summary

ComponentLifeTableInput.Parameter()

Method Summary

java.lang.String	**getCondition**() Getter for property condition.
java.lang.String	**getName**() Getter for property name.
java.util.Vector	**getTable**() Getter for property table.
java.util.Vector	**getVariables**() Getter for property variables.
void	**setCondition**(java.lang.String condition) Setter for property condition.
void	**setName**(java.lang.String name) Setter for property name.

void	**setTable**(java.util.Vector table) Setter for property table.
void	**setVariables**(java.util.Vector variables) Setter for property variables.

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

ComponentLifeTableInput.Parameter

public **ComponentLifeTableInput.Parameter**()

Method Detail

getName

public java.lang.String **getName**()

Getter for property name.

Returns:

Value of property name.

---

setName

public void **setName**(java.lang.String name)

Setter for property name.

Parameters:

name - New value of property name.

---

getCondition

public java.lang.String **getCondition**()

Getter for property condition.

Returns:

Value of property condition.

---

setCondition

`public void setCondition(java.lang.String condition)`

Setter for property condition.

Parameters:

`condition` - New value of property condition.

---

getTable

`public java.util.Vector getTable()`

Getter for property table.

Returns:

Value of property table.

---

setTable

`public void setTable(java.util.Vector table)`

Setter for property table.

Parameters:

`table` - New value of property table.

---

getVariables

`public java.util.Vector getVariables()`

Getter for property variables.

Returns:

Value of property variables.

---

setVariables

`public void setVariables(java.util.Vector variables)`

Setter for property variables.

Parameters:

`variables` - New value of property variables.

Class ComponentLifeUserInput

```
java.lang.Object
  ComponentLifeUserInput
```

All Implemented Interfaces:

java.io.Serializable

public class **ComponentLifeUserInput**

extends java.lang.Object

implements java.io.Serializable

This class represents the set of input paramaters and their associated values.

Currently objects from this class are created without the use of graphical user interface within the main method of ComponentLifeSituatedCBR.

See Also:

Serialized Form

Constructor Summary

ComponentLifeUserInput() Creates a new instance of ComponentLifeUserInput
ComponentLifeUserInput(ComponentLifeUserInput oldInput) Duplicates a new instance of ComponentLifeUserInput
ComponentLifeUserInput(double locX, double locY, java.lang.String material, java.lang.String componentType, java.lang.String maintenanceState, java.lang.String cleaningState, java.lang.String cleaningCondition, java.lang.String locationInBuilding, java.lang.String geoLocationCondition) Creates a new instance of ComponentLifeUserInput

Method Summary

java.lang.String	**getCleaningCondition**() Getter for property cleaningCondition.
java.lang.String	**getCleaningState**()

	Getter for property cleaningState.
java.lang.String	**getComponentType**() Getter for property componentType.
java.lang.String	**getGeoLocationCondition**() Getter for property geoLocationState.
java.lang.String	**getLocationInBuilding**() Getter for property locationInBuilding.
java.lang.String	**getMaintenanceState**() Getter for property maintanenceState.
java.lang.String	**getMaterial**() Getter for property material.
double	**getSalt**() Getter for property salt.
ComponentLifeGeoLocation	**getSiteLocation**() Getter for property geoLocation.
double	**getToW**() Getter for property toW.
void	**setCleaningCondition**(java.lang.String cleaningCondition) Setter for property cleaningCondition.
void	**setCleaningState**(java.lang.String cleaningState) Setter for property cleaningState.
void	**setComponentType**(java.lang.String componentType) Setter for property componentType.
void	**setGeoLocationCondition**(java.lang.String geoLocationConditi Setter for property geoLocationState.
void	**setLocationInBuilding**(java.lang.String locationInBuilding) Setter for property locationInBuilding.
void	**setMaintenanceState**(java.lang.String maintenanceState) Setter for property maintanenceState.
void	**setMaterial**(java.lang.String material) Setter for property material.
void	**setSalt**() Setter for property salt using a ComponentLifeHolisticModel object
void	**setSalt**(double salt) Setter for property salt.
void	**setSiteLocation**(ComponentLifeGeoLocation geoLocation) Setter for property geoLocation.
void	**setToW**() Setter for property toW using a ComponentLifeHolisticModel object

void	**setToW**(double toW) Setter for property toW.
java.lang.String	**toString**() Return a String representation of the object.

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

ComponentLifeUserInput

```
public ComponentLifeUserInput()
```

Creates a new instance of ComponentLifeUserInput

ComponentLifeUserInput

```
public ComponentLifeUserInput(double locX,
                              double locY,
                              java.lang.String material,
                              java.lang.String componentType,
                              java.lang.String maintenanceState,
                              java.lang.String cleaningState,
                              java.lang.String cleaningCondition,
                              java.lang.String locationInBuilding,
                              java.lang.String geoLocationCondition)
```

Creates a new instance of ComponentLifeUserInput

ComponentLifeUserInput

```
public ComponentLifeUserInput(ComponentLifeUserInput oldInput)
```

Duplicates a new instance of ComponentLifeUserInput

Method Detail

getSiteLocation

```
public ComponentLifeGeoLocation getSiteLocation()
```

Getter for property geoLocation.

Returns:

Value of property geoLocation.

---

setSiteLocation

`public void` **`setSiteLocation`**`(ComponentLifeGeoLocation geoLocation)`

Setter for property geoLocation.

Parameters:

`geoLocation` - New value of property geoLocation.

---

getMaterial

`public java.lang.String` **`getMaterial`**`()`

Getter for property material.

Returns:

Value of property material.

---

setMaterial

`public void` **`setMaterial`**`(java.lang.String material)`

Setter for property material.

Parameters:

`material` - New value of property material.

---

getComponentType

`public java.lang.String` **`getComponentType`**`()`

Getter for property componentType.

Returns:

Value of property componentType.

---

setComponentType

`public void` **`setComponentType`**`(java.lang.String componentType)`

Setter for property componentType.

Parameters:

`componentType` - New value of property componentType.

---

getMaintenanceState

`public java.lang.String` **`getMaintenanceState`**`()`

Getter for property maintanenceState.

Returns:

Value of property maintanenceState.

---

setMaintenanceState

`public void` **`setMaintenanceState`**`(java.lang.String maintenanceState)`

Setter for property maintanenceState.

---

getCleaningState

`public java.lang.String` **`getCleaningState`**`()`

Getter for property cleaningState.

Returns:

Value of property cleaningState.

---

setCleaningState

`public void` **`setCleaningState`**`(java.lang.String cleaningState)`

Setter for property cleaningState.

Parameters:

`cleaningState` - New value of property cleaningState.

---

getCleaningCondition

`public java.lang.String` **`getCleaningCondition`**`()`

Getter for property cleaningCondition.

Returns:

Value of property cleaningCondition.

---

setCleaningCondition

`public void` **`setCleaningCondition`**`(java.lang.String cleaningCondition)`

Setter for property cleaningCondition.

Parameters:

`cleaningCondition` - New value of property cleaningCondition.

---

getLocationInBuilding

`public java.lang.String` **`getLocationInBuilding`**`()`

Getter for property locationInBuilding.

Returns:

Value of property locationInBuilding.

---

setLocationInBuilding

`public void` **`setLocationInBuilding`**`(java.lang.String locationInBuilding)`

Setter for property locationInBuilding.

Parameters:

`locationInBuilding` - New value of property locationInBuilding.

---

getGeoLocationCondition

```
public java.lang.String getGeoLocationCondition()
```

Getter for property geoLocationState.

Returns:

Value of property geoLocationState.

---

setGeoLocationCondition

```
public void setGeoLocationCondition(java.lang.String geoLocationCondition)
```

Setter for property geoLocationState.

---

getToW

```
public double getToW()
```

Getter for property toW.

Returns:

Value of property toW.

---

setToW

```
public void setToW(double toW)
```

Setter for property toW.

Parameters:

`toW` - New value of property toW.

---

setToW

```
public void setToW()
```

Setter for property toW using a ComponentLifeHolisticModel object.

---

getSalt

```
public double getSalt()
```

Getter for property salt.

Returns:

Value of property salt.

---

setSalt

```
public void setSalt(double salt)
```

Setter for property salt.

Parameters:

`salt` - New value of property salt.

---

setSalt

`public void` **`setSalt`**`()`

Setter for property salt using a ComponentLifeHolisticModel object.

---

toString

`public java.lang.String` **`toString`**`()`

Return a String representation of the object.

Appendix V Codes for Unit Testing (for class ComponentLifeTableInput)

```
public static void main(String[] args){
    System.out.println("\n\n\n******** Testing of ComponentLifeTableInput Class ********\n\n\n");
    ComponentLifeTableInput clife = new ComponentLifeTableInput("D:\\My Documents\\Working\\CRC_SituatedCaseBasedSystem\\Coding\\userInputCFG.txt");
    Vector cont = clife.getTableContents();

    /* Testing cods to look into the table
    int size = cont.size();
    for (int i=0; i<size; i++){
Parameter par = (Parameter)cont.get(i);

        System.out.println("Name: " + par.getName());
        System.out.println("Condition: " + par.getCondition());
        Vector var = par.getVariables();
        int sizeV = var.size();
        for (int j=0; j<sizeV; j++){
            System.out.println("\tVariables: " + ((String)var.get(j)));
        }
        Vector tab = par.getTable();
        int sizeT = tab.size();
        for (int k=0; k<sizeT; k++){
            System.out.println("\tTable: " + ((String)tab.get(k)));
        }
        System.out.println("\n");
    }*/

    //String col = "0 0.7 1   0.9";
    //System.out.println("Test: " + clife.getColumn(col, 3));

    //System.out.println("Test: " + clife.getParameterSimilarityIndex("maintenance state", "maintained", "maintained", "NULL"));
    //System.out.println("Test: " + clife.getParameterSimilarityIndex("maintenance state", "not maintained", "not maintained", "NULL"));
    //System.out.println("Test: " + clife.getParameterSimilarityIndex("maintenance state", "maintained", "not maintained", "NULL"));
    //System.out.println("Test: " + clife.getParameterSimilarityIndex("maintenance state", "not maintained", "maintained", "NULL"));

    //System.out.println("Test: " + clife.getParameterSimilarityIndex("cleaning", "cleaned", "cleaned", "gunk can collect"));
    //System.out.println("Test: " + clife.getParameterSimilarityIndex("cleaning", "not cleaned", "not cleaned", "gunk can collect"));
    //System.out.println("Test: " + clife.getParameterSimilarityIndex("cleaning", "cleaned", "not cleaned", "gunk can collect"));
    //System.out.println("Test: " + clife.getParameterSimilarityIndex("cleaning", "not cleaned", "cleaned", "gunk can collect"));

    //System.out.println("Test: " + clife.getParameterSimilarityIndex("cleaning", "cleaned", "cleaned", "gunk cannot collect"));
    //System.out.println("Test: " + clife.getParameterSimilarityIndex("cleaning", "not cleaned", "not cleaned", "gunk cannot collect"));
    //System.out.println("Test: " + clife.getParameterSimilarityIndex("cleaning", "cleaned", "not cleaned", "gunk cannot collect"));
    //System.out.println("Test: " + clife.getParameterSimilarityIndex("cleaning", "not cleaned", "cleaned", "gunk cannot collect"));
    /*
    System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "open rooftop", "open rooftop", "NULL"));
    System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "open rooftop", "open wall", "NULL"));
    System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "open rooftop", "sheltered wall", "NULL"));
    System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "open rooftop", "edges and external corners of walls or roofs", "NULL"));
    System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "open rooftop", "dirt accumulation zone", "NULL"));
    System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "open rooftop", "roof cavity", "NULL"));
    System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "open rooftop", "wall cavity", "NULL"));
    System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "open rooftop", "moisture accumulation points in wall cavities", "NULL"));
    System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "open rooftop", "underfloor cavity", "NULL"));
    System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "open rooftop", "underfloor positions in contact with earth", "NULL"));
```

```
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "open rooftop", "semi enclosed space", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "open rooftop", "enclosed room", "NULL"));
*/
/*
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "open wall", "open rooftop", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "open wall", "open wall", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "open wall", "sheltered wall", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "open wall", "edges and external corners of walls or roofs", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "open wall", "dirt accumulation zone", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "open wall", "roof cavity", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "open wall", "wall cavity", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "open wall", "moisture accumulation points in wall cavities", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "open wall", "underfloor cavity", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "open wall", "underfloor positions in contact with earth", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "open wall", "semi enclosed space", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "open wall", "enclosed room", "NULL"));
*/
/*
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "sheltered wall", "open rooftop", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "sheltered wall", "open wall", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "sheltered wall", "sheltered wall", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "sheltered wall", "edges and external corners of walls or roofs", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "sheltered wall", "dirt accumulation zone", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "sheltered wall", "roof cavity", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "sheltered wall", "wall cavity", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "sheltered wall", "moisture accumulation points in wall cavities", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "sheltered wall", "underfloor cavity", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "sheltered wall", "underfloor positions in contact with earth", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "sheltered wall", "semi enclosed space", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "sheltered wall", "enclosed room", "NULL"));
*/
/*
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "edges and external corners of walls or roofs", "open rooftop", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "edges and external corners of walls or roofs", "open wall", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "edges and external corners of walls or roofs", "sheltered wall", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "edges and external corners of walls or roofs", "edges and external corners of walls or roofs", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "edges and external corners of walls or roofs", "dirt accumulation zone", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "edges and external corners of walls or roofs", "roof cavity", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "edges and external corners of walls or roofs", "wall cavity", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "edges and external corners of walls or roofs", "moisture accumulation points in wall cavities", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "edges and external corners of walls or roofs", "underfloor cavity", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "edges and external corners of walls or roofs", "underfloor positions in contact with earth", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "edges and external corners of walls or roofs", "semi enclosed space", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "edges and external corners of walls or roofs", "enclosed room", "NULL"));
*/
/*
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "dirt accumulation zone", "open rooftop", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "dirt accumulation zone", "open wall", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "dirt accumulation zone", "sheltered wall", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "dirt accumulation zone", "edges and external corners of walls or roofs", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "dirt accumulation zone", "dirt accumulation zone", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "dirt accumulation zone", "roof cavity", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "dirt accumulation zone", "wall cavity", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "dirt accumulation zone", "moisture accumulation points in wall cavities", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "dirt accumulation zone", "underfloor cavity", "NULL"));
```

```
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "dirt accumulation zone", "underfloor positions in contact with earth", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "dirt accumulation zone", "semi enclosed space", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "dirt accumulation zone", "enclosed room", "NULL"));
*/
/*
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "roof cavity", "open rooftop", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "roof cavity", "open wall", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "roof cavity", "sheltered wall", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "roof cavity", "edges and external corners of walls or roofs", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "roof cavity", "dirt accumulation zone", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "roof cavity", "roof cavity", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "roof cavity", "wall cavity", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "roof cavity", "moisture accumulation points in wall cavities", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "roof cavity", "underfloor cavity", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "roof cavity", "underfloor positions in contact with earth", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "roof cavity", "semi enclosed space", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "roof cavity", "enclosed room", "NULL"));
*/
/*
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "wall cavity", "open rooftop", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "wall cavity", "open wall", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "wall cavity", "sheltered wall", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "wall cavity", "edges and external corners of walls or roofs", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "wall cavity", "dirt accumulation zone", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "wall cavity", "roof cavity", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "wall cavity", "wall cavity", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "wall cavity", "moisture accumulation points in wall cavities", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "wall cavity", "underfloor cavity", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "wall cavity", "underfloor positions in contact with earth", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "wall cavity", "semi enclosed space", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "wall cavity", "enclosed room", "NULL"));
*/
/*
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "moisture accumulation points in wall cavities", "open rooftop", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "moisture accumulation points in wall cavities", "open wall", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "moisture accumulation points in wall cavities", "sheltered wall", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "moisture accumulation points in wall cavities", "edges and external corners of walls or roofs", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "moisture accumulation points in wall cavities", "dirt accumulation zone", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "moisture accumulation points in wall cavities", "roof cavity", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "moisture accumulation points in wall cavities", "wall cavity", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "moisture accumulation points in wall cavities", "moisture accumulation points in wall cavities", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "moisture accumulation points in wall cavities", "underfloor cavity", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "moisture accumulation points in wall cavities", "underfloor positions in contact with earth", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "moisture accumulation points in wall cavities", "semi enclosed space", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "moisture accumulation points in wall cavities", "enclosed room", "NULL"));
*/
/*
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "underfloor cavity", "open rooftop", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "underfloor cavity", "open wall", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "underfloor cavity", "sheltered wall", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "underfloor cavity", "edges and external corners of walls or roofs", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "underfloor cavity", "dirt accumulation zone", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "underfloor cavity", "roof cavity", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "underfloor cavity", "wall cavity", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "underfloor cavity", "moisture accumulation points in wall cavities", "NULL"));
```

```
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "underfloor cavity", "underfloor cavity", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "underfloor cavity", "underfloor positions in contact with earth", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "underfloor cavity", "semi enclosed space", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "underfloor cavity", "enclosed room", "NULL"));
*/
/*
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "underfloor positions in contact with earth", "open rooftop", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "underfloor positions in contact with earth", "open wall", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "underfloor positions in contact with earth", "sheltered wall", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "underfloor positions in contact with earth", "edges and external corners of walls or roofs", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "underfloor positions in contact with earth", "dirt accumulation zone", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "underfloor positions in contact with earth", "roof cavity", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "underfloor positions in contact with earth", "wall cavity", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "underfloor positions in contact with earth", "moisture accumulation points in wall cavities", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "underfloor positions in contact with earth", "underfloor cavity", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "underfloor positions in contact with earth", "underfloor positions in contact with earth", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "underfloor positions in contact with earth", "semi enclosed space", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "underfloor positions in contact with earth", "enclosed room", "NULL"));
*/
/*
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "semi enclosed space", "open rooftop", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "semi enclosed space", "open wall", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "semi enclosed space", "sheltered wall", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "semi enclosed space", "edges and external corners of walls or roofs", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "semi enclosed space", "dirt accumulation zone", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "semi enclosed space", "roof cavity", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "semi enclosed space", "wall cavity", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "semi enclosed space", "moisture accumulation points in wall cavities", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "semi enclosed space", "underfloor cavity", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "semi enclosed space", "underfloor positions in contact with earth", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "semi enclosed space", "semi enclosed space", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "semi enclosed space", "enclosed room", "NULL"));
*/
/*
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "enclosed room", "open rooftop", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "enclosed room", "open wall", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "enclosed room", "sheltered wall", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "enclosed room", "edges and external corners of walls or roofs", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "enclosed room", "dirt accumulation zone", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "enclosed room", "roof cavity", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "enclosed room", "wall cavity", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "enclosed room", "moisture accumulation points in wall cavities", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "enclosed room", "underfloor cavity", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "enclosed room", "underfloor positions in contact with earth", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "enclosed room", "semi enclosed space", "NULL"));
System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "enclosed room", "enclosed room", "NULL"));
*/
// Out of range test
//System.out.println("Test: " + clife.getParameterSimilarityIndex("location in building", "enclosed rooms", "enclosed room", "NULL"));

//String str = "20 to 39";
//System.out.println("Max: " + (str.substring(str.indexOf("to")+2)).trim());
//System.out.println("Min: " + (str.substring(0,str.indexOf("to")-1)).trim());

//System.out.println("ToW: " + clife.getParameterSimilarityIndex("time of wetness", 0, 0, "marine application"));
```

```
//System.out.println("ToW: " + clife.getParameterSimilarityIndex("time of wetness", 5, 5, "marine application"));
//System.out.println("ToW: " + clife.getParameterSimilarityIndex("time of wetness", 19, 19, "marine application"));

//System.out.println("ToW: " + clife.getParameterSimilarityIndex("time of wetness", 20, 20, "marine application"));
//System.out.println("ToW: " + clife.getParameterSimilarityIndex("time of wetness", 25, 25, "marine application"));
//System.out.println("ToW: " + clife.getParameterSimilarityIndex("time of wetness", 39, 39, "marine application"));

//System.out.println("ToW: " + clife.getParameterSimilarityIndex("time of wetness", 40, 40, "marine application"));
//System.out.println("ToW: " + clife.getParameterSimilarityIndex("time of wetness", 45, 45, "marine application"));
//System.out.println("ToW: " + clife.getParameterSimilarityIndex("time of wetness", 59, 59, "marine application"));

//System.out.println("ToW: " + clife.getParameterSimilarityIndex("time of wetness", 60, 60, "marine application"));
//System.out.println("ToW: " + clife.getParameterSimilarityIndex("time of wetness", 65, 65, "marine application"));
//System.out.println("ToW: " + clife.getParameterSimilarityIndex("time of wetness", 79, 79, "marine application"));

//System.out.println("ToW: " + clife.getParameterSimilarityIndex("time of wetness", 80, 80, "marine application"));
//System.out.println("ToW: " + clife.getParameterSimilarityIndex("time of wetness", 85, 85, "marine application"));
//System.out.println("ToW: " + clife.getParameterSimilarityIndex("time of wetness", 100, 100, "marine application"));

//System.out.println("ToW: " + clife.getParameterSimilarityIndex("time of wetness", 5, 25, "marine application"));
//System.out.println("ToW: " + clife.getParameterSimilarityIndex("time of wetness", 5, 45, "marine application"));
//System.out.println("ToW: " + clife.getParameterSimilarityIndex("time of wetness", 5, 65, "marine application"));
//System.out.println("ToW: " + clife.getParameterSimilarityIndex("time of wetness", 5, 85, "marine application"));

//System.out.println("ToW: " + clife.getParameterSimilarityIndex("time of wetness", 25, 5, "marine application"));
//System.out.println("ToW: " + clife.getParameterSimilarityIndex("time of wetness", 25, 45, "marine application"));
//System.out.println("ToW: " + clife.getParameterSimilarityIndex("time of wetness", 25, 65, "marine application"));
//System.out.println("ToW: " + clife.getParameterSimilarityIndex("time of wetness", 25, 85, "marine application"));

//System.out.println("ToW: " + clife.getParameterSimilarityIndex("time of wetness", 45, 5, "marine application"));
//System.out.println("ToW: " + clife.getParameterSimilarityIndex("time of wetness", 45, 25, "marine application"));
//System.out.println("ToW: " + clife.getParameterSimilarityIndex("time of wetness", 45, 65, "marine application"));
//System.out.println("ToW: " + clife.getParameterSimilarityIndex("time of wetness", 45, 85, "marine application"));

//System.out.println("ToW: " + clife.getParameterSimilarityIndex("time of wetness", 65, 5, "marine application"));
//System.out.println("ToW: " + clife.getParameterSimilarityIndex("time of wetness", 65, 25, "marine application"));
//System.out.println("ToW: " + clife.getParameterSimilarityIndex("time of wetness", 65, 45, "marine application"));
//System.out.println("ToW: " + clife.getParameterSimilarityIndex("time of wetness", 65, 85, "marine application"));

//System.out.println("ToW: " + clife.getParameterSimilarityIndex("time of wetness", 85, 5, "marine application"));
//System.out.println("ToW: " + clife.getParameterSimilarityIndex("time of wetness", 85, 25, "marine application"));
//System.out.println("ToW: " + clife.getParameterSimilarityIndex("time of wetness", 85, 45, "marine application"));
//System.out.println("ToW: " + clife.getParameterSimilarityIndex("time of wetness", 85, 65, "marine application"));

//System.out.println("Salt: " + clife.getParameterSimilarityIndex("salinity factor", 0, 0, "marine application"));
//System.out.println("Salt: " + clife.getParameterSimilarityIndex("salinity factor", 2, 2, "marine application"));
//System.out.println("Salt: " + clife.getParameterSimilarityIndex("salinity factor", 4, 4, "marine application"));

//System.out.println("Salt: " + clife.getParameterSimilarityIndex("salinity factor", 5, 5, "marine application"));
//System.out.println("Salt: " + clife.getParameterSimilarityIndex("salinity factor", 10, 10, "marine application"));
//System.out.println("Salt: " + clife.getParameterSimilarityIndex("salinity factor", 15, 15, "marine application"));

//System.out.println("Salt: " + clife.getParameterSimilarityIndex("salinity factor", 16, 16, "marine application"));
//System.out.println("Salt: " + clife.getParameterSimilarityIndex("salinity factor", 28, 28, "marine application"));
//System.out.println("Salt: " + clife.getParameterSimilarityIndex("salinity factor", 40, 40, "marine application"));
```

```
        //System.out.println("Salt: " + clife.getParameterSimilarityIndex("salinity factor", 41, 41, "marine application"));
        //System.out.println("Salt: " + clife.getParameterSimilarityIndex("salinity factor", 70, 70, "marine application"));
        //System.out.println("Salt: " + clife.getParameterSimilarityIndex("salinity factor", 100, 100, "marine application"));

        //System.out.println("Salt: " + clife.getParameterSimilarityIndex("salinity factor", 101, 101, "marine application"));
        //System.out.println("Salt: " + clife.getParameterSimilarityIndex("salinity factor", 200, 200, "marine application"));
        //System.out.println("Salt: " + clife.getParameterSimilarityIndex("salinity factor", 300, 300, "marine application"));

        //System.out.println("Salt: " + clife.getParameterSimilarityIndex("salinity factor", 301, 301, "marine application"));
        //System.out.println("Salt: " + clife.getParameterSimilarityIndex("salinity factor", 350, 350, "marine application"));
        //System.out.println("Salt: " + clife.getParameterSimilarityIndex("salinity factor", 400, 400, "marine application"));

        //System.out.println("Salt: " + clife.getParameterSimilarityIndex("salinity factor", 2, 10, "marine application"));
        //System.out.println("Salt: " + clife.getParameterSimilarityIndex("salinity factor", 2, 28, "marine application"));
        //System.out.println("Salt: " + clife.getParameterSimilarityIndex("salinity factor", 2, 70, "marine application"));
        //System.out.println("Salt: " + clife.getParameterSimilarityIndex("salinity factor", 2, 200, "marine application"));
        //System.out.println("Salt: " + clife.getParameterSimilarityIndex("salinity factor", 2, 350, "marine application"));

        //System.out.println("Salt: " + clife.getParameterSimilarityIndex("salinity factor", 10, 2, "marine application"));
        //System.out.println("Salt: " + clife.getParameterSimilarityIndex("salinity factor", 10, 28, "marine application"));
        //System.out.println("Salt: " + clife.getParameterSimilarityIndex("salinity factor", 10, 70, "marine application"));
        //System.out.println("Salt: " + clife.getParameterSimilarityIndex("salinity factor", 10, 200, "marine application"));
        //System.out.println("Salt: " + clife.getParameterSimilarityIndex("salinity factor", 10, 350, "marine application"));

        //System.out.println("Salt: " + clife.getParameterSimilarityIndex("salinity factor", 28, 2, "marine application"));
        //System.out.println("Salt: " + clife.getParameterSimilarityIndex("salinity factor", 28, 10, "marine application"));
        //System.out.println("Salt: " + clife.getParameterSimilarityIndex("salinity factor", 28, 70, "marine application"));
        //System.out.println("Salt: " + clife.getParameterSimilarityIndex("salinity factor", 28, 200, "marine application"));
        //System.out.println("Salt: " + clife.getParameterSimilarityIndex("salinity factor", 28, 350, "marine application"));

        //System.out.println("Salt: " + clife.getParameterSimilarityIndex("salinity factor", 70, 2, "marine application"));
        //System.out.println("Salt: " + clife.getParameterSimilarityIndex("salinity factor", 70, 10, "marine application"));
        //System.out.println("Salt: " + clife.getParameterSimilarityIndex("salinity factor", 70, 28, "marine application"));
        //System.out.println("Salt: " + clife.getParameterSimilarityIndex("salinity factor", 70, 200, "marine application"));
        //System.out.println("Salt: " + clife.getParameterSimilarityIndex("salinity factor", 70, 350, "marine application"));

        //System.out.println("Salt: " + clife.getParameterSimilarityIndex("salinity factor", 200, 2, "marine application"));
        //System.out.println("Salt: " + clife.getParameterSimilarityIndex("salinity factor", 200, 10, "marine application"));
        //System.out.println("Salt: " + clife.getParameterSimilarityIndex("salinity factor", 200, 28, "marine application"));
        //System.out.println("Salt: " + clife.getParameterSimilarityIndex("salinity factor", 200, 70, "marine application"));
        //System.out.println("Salt: " + clife.getParameterSimilarityIndex("salinity factor", 200, 350, "marine application"));

        //System.out.println("Salt: " + clife.getParameterSimilarityIndex("salinity factor", 350, 2, "marine application"));
        //System.out.println("Salt: " + clife.getParameterSimilarityIndex("salinity factor", 350, 10, "marine application"));
        //System.out.println("Salt: " + clife.getParameterSimilarityIndex("salinity factor", 350, 28, "marine application"));
        //System.out.println("Salt: " + clife.getParameterSimilarityIndex("salinity factor", 350, 70, "marine application"));
        //System.out.println("Salt: " + clife.getParameterSimilarityIndex("salinity factor", 350, 200, "marine application"));

    }
```

12. AUTHOR BIOGRAPHIES