NTIS #PB97-142848

SSC-395

SHIP MAINTENANCE PROJECT

Phases II and III- Volume 3

Repair Management System for Critical Structural Details in Ships

SHIP STRUCTURE COMMITTEE

1997

c1

SHIP STRUCTURE COMMITTEE

The SHIP STRUCTURE COMMITTEE is constituted to prosecute a research program to improve the hull structures of ships and other marine structures by an extension of knowledge pertaining to design, materials, and methods of construction.

RADM J. C. Card, USCG (Chairman)
Chief, Office of Marine Safety, Security and Environmental Protection
U. S. Coast Guard

Mr. John Grinstead
Director, Policy and Legislation
Marine Regulatory Directorate
Transport Canada

Mr. Edwin B. Schimler
Associate Administrator for Shipbuilding and Technology Development
Maritime Administration

Dr. Donald Liu
Senior Vice President
American Bureau of Shipping

Mr. Robert McCarthy
Director, Survivability and Structural Integrity Group (SEA O3P)
Naval Sea Systems Command

Mr. Thomas Connors
Acting Director of Engineering (N7)
Military Sealift Command

Dr. Ross Grahm
Head, Hydronautics Section
Defence Research Establishment-Atlantic

EXECUTIVE DIRECTOR

CDR Stephen E. Sharpe, USCG
U. S. Coast Guard

CONTRACTING OFFICER TECHNICAL REPRESENTATIVE

Mr. William J. Siekierka
Naval Sea Systems Command

SHIP STRUCTURE SUBCOMMITTEE

The SHIP STRUCTURE SUBCOMMITTEE acts for the Ship Structure Committee on technical matters by providing technical coordination for determinating the goals and objectives of the program and by evaluating and interpreting the results in terms of structural design, construction, and operation.

MILITARY SEALIFT COMMAND

Mr. Robert E. Van Jones (Chairman)
Mr. Rickard A. Anderson
Mr. Michael W. Touma
Mr. Jeffrey E. Beach

MARITIME ADMINISTRATION

Mr. Frederick Seibold
Mr. Richard P. Voelker
Mr. Chao H. Lin
Dr. Walter M. Maclean

U. S. COAST GUARD

CAPT George Wright
Mr. Walter Lincoln
Mr. Rubin Sheinberg

AMERICAN BUREAU OF SHIPPING

Mr. Glenn Ashe
Mr. John F. Conlon
Mr. Phillip G. Rynn
Mr. William Hanzalek

NAVAL SEA SYSTEMS COMMAND

Mr. W. Thomas Packard
Mr. Charles L. Null
Mr. Edward Kadala
Mr. Allen H. Engle

TRANSPORT CANADA

Mr. Peter Timonin
Mr. Felix Connolly
Mr. Francois Lamanque

DEFENCE RESEARCH ESTABLISHMENT ATLANTIC

Dr. Neil Pegg
LCDR Stephen Gibson
Dr. Roger Hollingshead
Mr. John Porter

SHIP STRUCTURE SUBCOMMITTEE LIAISON MEMBERS

SOCIETY OF NAVAL ARCHITECTS AND MARINE ENGINEERS
Dr. William Sandberg

CANADA CENTRE FOR MINERALS AND ENERGY TECHNOLOGIES
Dr. William R. Tyson

U. S. NAVAL ACADEMY
Dr. Ramswar Bhattacharyya

U. S. MERCHANT MARINE ACADEMY
Dr. C. B. Kim

U. S. COAST GUARD ACADEMY
CDR Bruce R. Mustain

U. S. TECHNICAL ADIVSORY GROUP TO THE INTERNATIONAL STANDARDS ORGANIZATION
CAPT Charles Piersall

AMERICAN WELDING SOCIETY
Mr. Richard French

NATIONAL ACADEMY OF SCIENCES - MARINE BOARD
Dr. Robert Sielski

NATIONAL ACADEMY OF SCIENCES - COMMITTEE ON MARINE STRUCTURES
Dr. John Landes

WELDING RESEARCH COUNCIL
Dr. Martin Prager

AMERICAN IRON AND STEEL INSTITUTE
Mr. Alexander D. Wilson

OFFICE OF NAVAL RESEARCH
Dr. Yapa D. S. Rajapaske

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
CAPT Alan J. Brown

STUDENT MEMBER
Mr. Jason Miller
Massachusetts Institute of Technology

c2

Member Agencies:

American Bureau of Shipping
Defence Research Establishment Atlantic
Maritime Administration
Military Sealift Command
Naval Sea Systems Command
Transport Canada
United States Coast Guard

Ship
Structure
Committee

An Interagency Advisory Committee

Address Correspondence to:

Executive Director
Ship Structure Committee
U.S. Coast Guard (G-MSE/SSC)
2100 Second Street, S.W.
Washington, D.C. 20593-0001
Ph: (202) 267-0003
Fax: (202) 267-4816

SSC-395
SR-1360
SR-1371

February 27, 1997

SHIP MAINTENANCE PROJECT
Phases II and III

This report presents the results of the second and third phases of the subject project of which phase one was first presented in our four volume set -- SSC-386. These studies investigated the development of engineering technology that could lead improvements in structural maintenance for new and existing tankers. These projects built further upon the work started in phase I specifically focusing on critical structural details and corrosion limits.

The report has been divided into five volumes, each of which may stand alone. Volume one opens with a summary of all three phases by Professor Robert G. Bea, the coordinating investigator for the program and follows with a report on corrosion limits for tankers. The second and fifth volumes look into evaluation of cracked critical structural details in tankers. The third volume presents theory and user instructions for software to manage repair of critical structural details. The fourth volume applies to fatigue classification of critical structural details. The software developed in the project will be available on the next Ship Structure Committee CD Rom release, which is anticipated to be released in the next year. The industry is encouraged to contact Professor Bea at the University of California, Berkeley to discuss further possibilities in application of the work undertaken here in the industry.

J. C. CARD
Rear Admiral, U.S. Coast Guard
Chairman, Ship Structure Committee

Technical Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
SSC-395-3	PB97-142848	

4. Title and Subtitle	5. Report Date
Ship Maintenance Project Phases II and III Volume 3 Repair Management System for Critical Structural Details in Ships	1997
	6. Performing Organization Code
7. Author(s) Robert Bea, Kai-tung Ma	8. Performing Organization Report No.
9. Performing Agency Name and Address University of California at Berkeley Department of Naval Architecture and Ocean Engineering Berkeley, CA 94720	10. Work Unit No. (TRAIS) 11. Contract or Grant No.
12. Sponsoring Agency Name and Address Ship Structure Committee U. S. Coast Guard (G-MSE/SSC) 2100 Second St. S.W. Washington, DC 21\0593-0001	13. Type of Report and Period Covered Final 14. Sponsoring Agency Code G-M

15. Supplementary Notes

Sponsored by the Ship Structure Committee. Jointly funded by other organizations as a joint industry project. See inside the report for futher details on sponsors.

16. Abstract

This report presents the results of the second and third phases of the subject project of which phase one was first presented in our four volume set - SSC-386. These studies investigated the development of engineering technology that could lead to improvements in structural maintenance for new and existing tankers. These projects built further upon the work started in phase I specifically focusing on critical structural details and corrosion limits.

The report has been divided into five volumes, each of which may stand alone. Volume one opens with a summary of all three phases by Professor Robert G. Bea, the coordinating investigator for the program, and follows with a report on corrosion limits for tankers. The second and fifth volumes look into evaluation of cracked critical structural details in tankers. The third volume presents theory and user instructions for software to manage repair of critical structural details. The fourth volume applies to fatigue classification of critical structural details. The software developed in the project will be available on the next Ship Structure Committee CD Rom release which is anticipated to be released in the next year. The industry is encouraged to contact Professor Bea at the University of California, Berkeley to discuss further possibilities in application of the work undertaken here in the industry.

17. Key Words	18. Distribution Statement
fatigue, critical structural details, tanker structures, repairs	Distribution unlimited, available from: National Technical Information Service U.S. Department of Commerce Springfield, VA 22151 (703)487-4690

19. Security Classif. (of this report)	20. SECURITY CLASSIF. (of this page)	21. No. of Pages	22. Price
Unclassified	Unclassified	350	$49.00-Paper

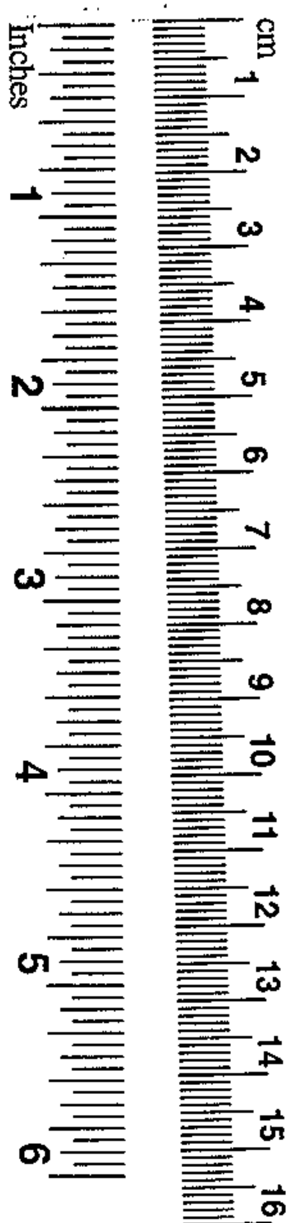
Form DOT F 1700.7 (8/72) Reproduction of form and completed page is authorized.

NIST
United States Department of Commerce
Technology Administration
National Institute of Standards and Technology
Metric Program, Gaithersburg, MD 20899

METRIC CONVERSION CARD

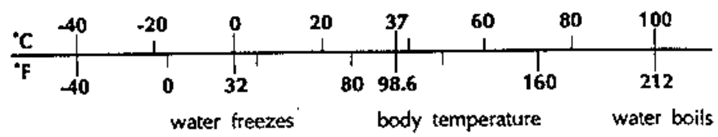
Approximate Conversions to Metric Measures

Symbol	When You Know	Multiply by	To Find	Symbol
	LENGTH			
in	inches	2.5	centimeters	cm
ft	feet	30	centimeters	cm
yd	yards	0.9	meters	m
mi	miles	1.6	kilometers	km
	AREA			
in^2	square inches	6.5	square centimeters	cm^2
ft^2	square feet	0.09	square meters	m^2
yd^2	square yards	0.8	square meters	m^2
mi^2	square miles	2.6	square kilometers	km^2
	acres	0.4	hectares	ha
	MASS (weight)			
oz	ounces	28	grams	g
lb	pounds	0.45	kilograms	kg
	short tons (2000 lb)	0.9	metric ton	t
	VOLUME			
tsp	teaspoons	5	milliliters	mL
Tbsp	tablespoons	15	milliliters	mL
in^3	cubic inches	16	milliliters	mL
fl oz	fluid ounces	30	milliliters	mL
c	cups	0.24	liters	L
pt	pints	0.47	liters	L
qt	quarts	0.95	liters	L
gal	gallons	3.8	liters	L
ft^3	cubic feet	0.03	cubic meters	m^3
yd^3	cubic yards	0.76	cubic meters	m^3
	TEMPERATURE (exact)			
°F	degrees Fahrenheit	subtract 32, multiply by 5/9	degrees Celsius	°C



Approximate Conversions from Metric Measures

Symbol	When You Know	Multiply by	To Find	Symbol
	LENGTH			
mm	millimeters	0.04	inches	in
cm	centimeters	0.4	inches	in
m	meters	3.3	feet	ft
m	meters	1.1	yards	yd.
km	kilometers	0.6	miles	mi
	AREA			
cm^2	square centimeters	0.16	square inches	in^2
m^2	square meters	1.2	square yards	yd^2
km^2	square kilometers	0.4	square miles	mi^2
ha	hectares (10,000 m^2)	2.5	acres	
	MASS (weight)			
g	grams	0.035	ounces	oz
kg	kilograms	2.2	pounds	lb
t	metric ton (1,000 kg)	1.1	short tons	
	VOLUME			
mL	milliliters	0.03	fluid ounces	fl oz
mL	milliliters	0.06	cubic inches	in^3
L	liters	2.1	pints	pt
L	liters	1.06	quarts	qt
L	liters	0.26	gallons	gal
m^3	cubic meters	35	cubic feet	ft^3
m^3	cubic meters	1.3	cubic yards	yd^3
	TEMPERATURE (exact)			
°C	degrees Celsius	multiply by 9/5, add 32	degrees Fahrenheit	°F



Ship Structural Maintenance Projects II and III
Cross Reference Listing

SSC Vol	SMP #	Title	Authors	Date	NTIS Number
	II				
2	-1	Fatigue Analysis of CSD in a 150K DWT Double-Hull Tanker	Xu, Bea	10/93	PB97-142830
2	-2	Fatigue Analysis of CSD in a 190K DWT Double-Hull Tanker	Xu, Bea	10/93	PB97-142830
2	-3	CSD Library and Finite Element Stress Contours	Xu, Bea	10/93	PB97-142830
1	-4	Development of a Rational Basis for Defining Corrosion Limits in Tankers	Mayoss, Bea	12/93	PB97-142822
3	-4a	RMS for CSD in Ships - User Manual	Ma, Bea	9/93	PB97-142848
3	-4b	RMS for CSD in Ships - Theory	Ma, Bea	9/93	PB97-142848
4		Fatigue Classification of CSD in Tankers	Schulte-Strathaus, Bea	1/94	PB97-142855
	III				
3	-1-1	RMS for Fatigue Cracks in Ship CSDs	Ma, Bea	10/94	PB97-142848
5	-2-1	Fitness for Purpose Analysis Procedure of Cracked CSDs in Tankers	Xu, Bea	1/95	PB97-142863
5	-2-2	A Load Shedding Model of Fracture Mechanics Analysis of Cracked SCDs in Tankers	Xu, Bea	1/95	PB97-142863
5	-2-3	FRACTURE- A Computer Code for Fracture Mechanics Analysis of Crack Growth of Cracked CSD in Tankers	Xu, Bea	1/95	PB97-142863
5	-5	Pro-IMR: A Computer Code for Probability-Based Inspection Planning	Xu, Bea	10/94	PB97-142863

Repair Management System for Critical Structural Details in Ships

Theory

Kai-tung Ma
and
Professor Robert G. Bea

Department of Naval Architecture and Offshore Engineering
University of California, Berkeley

7
8X

Table of Contents

11

List of Tables

Page

13
14X

List of Figures

Page

Repair Management System for Critical Structural Details in Ships

Theory

Currently, Study 4 is encountering problems in acquiring sufficient data on repairs and maintenance in order to carry out this study properly. In addition to this problem, there is a lack of presently available "qualified and motivated" research assistants. .... In generalizing the project's status to date, the study has progressed as well as possible with the limited amount of data available. The course that the study has been following has focused on the owner's point of view. Most of the current information being used for the ship summaries, verifications and repair/corrosion case studies has been obtained from the ship owners. In order for the project to continue using the current format and information available, all of the Project Technical Committee (PTC) members will have to provide more pertinent information on the details of the repair of the corrosion and fatigue failures (e.g. steel weights used, time of repair, effectiveness of the repair, more details on the location and repair method used). It seems that the problem with obtaining this information is that the pertinent data needed for this study is not readily accessible. This information must be located by the PTC members and forwarded in a timely manner.

In the second phase Mr. Keith Gallion (former employee of Newport News Shipbuilding), was recruited as the study GSR. He shifted the study concentration from a database development approach to a repair engineering approach. The goal was to develop and verify analytical tools for repair evaluations. A questionnaire was sent to the technical contacts in the SMP requesting input on the desired contents of the fatigue and corrosion repairs software in order to evaluate the needs of the marine industry. The highest priorities of participants that responded were the expected life analysis of repairs and a database of repair alternatives, Table 1.1. As a result, concentration in this research is placed on the development of these features within the RMS. The result of the questionnaire showed that a graphical database and associated expected life analysis of repair alternatives are desired. A FORTRAN program was developed by Gallion and Bea to help ship repair engineers evaluating fatigue repair life [Gallion and Bea, 1992]. The first generation program was named as **Repair Management System 1.0**. .

can be found. Fatigue cracks, corrosion, coating breakdowns and buckling are the most common failures. To fix these failures, there are three types of steel repairs: crack repair, steel renewal and steel reinforcement. Also there are three types of corrosion prevention: coating maintenance for general corrosion, maintenance for pitting/grooving as well as the maintenance of sacrificial anodes.

In short, maintenance involves three levels:

- **Inspections** to uncover structural problems.
- **Preventative maintenance** to address problems before they occur. This can include programs such as "just in time" coating maintenance to ensure wastage limits of plating are not exceeded.
- **Repair** of structural problems following discovery by inspection.

However the emphasis of this research is on the proper repair of critical structural detail (CSD) failures in ships. And the main focus of RMS is concentrated in the repair life estimation and database development of fatigue crack repair only. Here we only review one type of failure, fatigue cracks. Chapter 2 will discuss in detail the information on inspection, maintenance and repair of all types of failures.

Cracks are potentially the most serious of defects as they can grow rapidly leaving affected structure unable to bear loads. As the result of a crack, the structure around a crack must carry a greater loading that can in turn lead to its failure in the future. If this cracking process continues unchecked, hull girder or long large panels of side shell collapse can result. As a result, the ship structure has to be inspected periodically and repaired as warranted. Ship structure details can be grouped into two types according their importance in structural strength. Primary structure is the structure which contributes significantly to the main structural strength of the ship such as hull plates, stiffeners, principal decks, main transverses, and so on. Secondary structure is the structure which neither contributes to the structural strength nor the watertight integrity such as partition bulkheads, platforms and so on.

Cracks in primary structure may be temporarily repaired by fitting double plates or gouging out the crack and filling in with weld metal. Gouging and re-welding is an easy and common means of repair. However, the strength of re-welded CSD is, almost invariably, worse than the original CSD. The repaired plate and/or weld will create new crack potentials and thus may fail even earlier. The better way of repair is to modify the local geometry to reduce the stress concentrations. Such repairs are sometimes considered in attempting to get the ship to a facility where full repairs can be made. If a longer life continuance is expected for the ship, a more robust repair such as design modification should be considered.

In the other hand, cracks in secondary structure may be arrested temporarily by drilling a hole of diameter equal to the plate thickness at a distance of two plate thicknesses in front of the visible crack tip and on a line with the direction of anticipated crack propagation.

It is difficult to decide which repair method is most reliable and cost effective for a particular crack. The selection of different repair alternatives depends on the location of the crack and the expected life continuance of the ship. In Chapter 4, we will discuss to select a repair alternative and the decision making process.

1.3 Objectives

Through experience, more advanced design procedures and tougher materials, modern ships usually don't suffer catastrophic failures. More freqently, they are plagued with the less dramatic problems of localized structural failures.

It was the goal of this research to review the process of structural maintenance and repair of crude oil carriers and to investigate a new approach to help manage the information used to make good repair decisions. Specifically the project was intended to develop a practical tool for fatigue crack repairs to help improve the durability of existing ships.

Recently, considerable effort has been put into understanding the effectiveness of specific repairs, especially those associated with fatigue of CSDs. This effort has resulted both from an aging fleet of existing ships and a heightened public interest in environmental issues [USCG, 1990][Jordon, 1978, 1980][TSCF, 1991]. In addition, records of ship condition are shifting from paper-based systems to computerized systems that contain inspection and repair information in a database format. This information about ship maintenance and repair can be sorted by an experienced

CHAPTER 1. BACKGROUND AND INTRODUCTION

1.1 Background

This report is one of the two final reports of a research project "Repair Management System for Critical Structural Details in Ships (RMS)." The other one is the user manual for RMS 2.0. This report contains the source code. The project was conducted during the period October 1, 1992 to September 30, 1993. It was carried out in the department of Naval Architecture and Offshore Engineering (NAOE), University of California at Berkeley. The Graduate Student Researcher, Kai-tung Ma, performed research under the supervision of the principal investigator, Professor Robert G. Bea. The following three organizations sponsored this research project:

- Arco Marine Incorporated.,
- Lisnave - Estaleiros Navais De Lisboa S.A.,
- Ship Structure Committee.

The RMS project developed as the result of a two year Joint Industry Research Project "Structural Maintenance Project for New and Existing Ships (SMP)." The SMP was conducted during the period 1990 - 1992 with the participation of 22 organizations by the Department of NAOE, University of California at Berkeley. It included six related studies. The study, Fatigue and Corrosion Repair Assessments, resulted in the RMS 1.0 and was the predecessor of the RMS 2.0.

Before introducing the basics of ship repair and the objective of RMS, it is worth reviewing the background of SMP. This will summarize the problems that have been confronted in the pervious study and the progress developed in the first phase of the work.

Two GSR's worked on the initial development for the first year, Mr. Bob Baker and Mr. Martin Cepauskas. The following is a summary of their findings.

(5) utilizing historical ship data, and
(6) utilizing both numeric and symbolic information.

1.5 Contents of Report

In Chapter 2 the basics of ship structural inspection and maintenance are discussed. These basics include an introduction of inspection programs, crack repair, steel renewal due to corrosion or buckling, pitting and grooving.

In Chapter 3 the RMS approach is discussed. Four basic steps in determining the best repair are summarized first. Then the modules of the RMS system are introduced. Details of a computer implementation of a complete RMS to analyze the mode and cause of failure, select repair alternatives, evaluated the life of the alternatives, and perform a decision analysis on these alternatives are discussed.

In Chapter 4 the RMS repair alternative selection analysis is outlined. The general strategies for crack repair is outlined. The RMS repair alternative selection is discussed in detail on some of the most critical structural details like beam brackets, flatbar stiffeners and longitudinal cutouts. The specifics of CSD repair are discussed. In addition, the repair decision making is discussed.

In Chapter 5 a method for simplified comparative analysis is proposed to estimate the fatigue lives of the repair alternatives. Several considerations are discussed including cumulative fatigue damage model, SN curve considerations, stress concentration factor considerations and Weibull loading model. The procedures of computing repair life are outlined. Also an example of repair alternative evaluation is reviewed.

In Chapter 6 the RMS crack repair database for the fatigue mode of structural failure is outlined.

In Chapter 7 the RMS approach is used in the development of a computer program to illustrate the evaluation of repair alternatives for fatigue failure of some CSDs. A case study analysis is conducted to verify the code and illustrate its effectiveness as a repair tool.

Finally, in Chapter 8 the research is summarized with some concluding remarks and recommendations for future developments.

1.6 Acknowledgment

The author wishes to express his appreciation to those individuals and organizations who have made significant contributions to this report. Arco Marine Inc., Lisnave and the Ship Structure Committee sponsored this project. Keith Gallion helped begin this research. Professor R. G. Bea helped initiate and conduct this project.

this goal, the approach taken by RMS is to provide intelligent front-end access to the information required to make repair decisions.

The RMS project combined the use of experience-based knowledge of side shell critical structural details (CSDs) and simplified analytical procedures in order to rank repair alternatives according to the expected life and cost of the repair. The user must select the most appropriate alternative from his or her knowledge of the economics of the ship. For example, for a fracture which took ten years to develop and discover, the repair options might be:

(1) Grind out crack and re-weld--5 years expected life
(2) Cut out section and butt-weld new piece--10 years expected life
(3) Add one bracket --12 years expected life
(4) Add two brackets --15 years expected life

Depending on the economic goals of the owner, a different repair alternative will be selected. For example, if the ship has only two more years in service, the cheapest alternative with an expected life of greater than two years will be selected.

The approach taken in this research was to expand these initial efforts to make the system more powerful and effective in promoting intelligent repair decisions. Areas of improvement and enhancement included:

(1) Addition of more CSDs to the capabilities of the system.
(2) Enhancement of graphical capabilities of the system.
(3) Enhancement of approach used for life estimates and economic considerations.

While including and developing the above features, the functions and advantages of the RMS were intended to be:

(1) providing a consistent repair strategy,
(2) ensuring more complete evaluation in timely manner,
(3) increasing level of expertise in the shipyard and office,
(4) promoting sharing of repair information among ship owners, operators and shipyards,

(3) perform a case study using the developed tool for a side shell critical structural detail.

The project was intended to enhance and modify the capabilities of the Repair Management System (RMS)--a computer system to aid in the diagnosis of ship (especially tanker) structural fatigue and corrosion failures and the prescription of the best repair alternative.

1.4 RMS Approach

When a structural failure in the form of cracking is discovered by inspection, a decision must be made as to the most effective repair. This decision is difficult due to the vast array of engineering, construction and repair knowledge. However, many additional factors must also be considered in a much shorter time. These factors include technical, economic, and logistic factors. As a result of the complexity and the short time allowed, the ship repair currently relies heavily on the experience of repair engineers and repair yard personnel. There is simply not enough time to take into account all possible factors and perform detailed analyses. Repair decisions often lack thorough technical and economic evaluation, but serve to get ships back into service quickly.

The repair of ships may be separated into two approaches. These are:

(1) Traditional Experience-Based Approach--repair decisions made based on experience. Decisions are made quickly, but little technical basis for some decisions due to complexity of the problems. No detailed analysis involved.
(2) Detailed Analysis Approach--lengthy detailed analysis conducted to resolve particularly troublesome repair problems. Analysis involves detailed ship motion analysis, global and local finite element models, and fatigue analysis. This approach is rarely used.

Clearly, the traditional approach lacks adequate technical justification and the detailed approach, although necessary at times, is inadequate to make on-the-spot repair decisions. The goal of RMS is to provide a computerized system to allow a more complete evaluation of the repair alternatives in a reasonable time period. In order to accomplish maintenance and repair can be sorted by an experienced repair engineer to help evaluate the effectiveness of past repairs and assess the overall condition of the ship.

The most technical part of the ship maintenance and repair is the decision making on choosing a suitable and reliable repair method for a particular structural failure. Ship structural repair decisions are difficult due to the vast array of engineering knowledge which must be assimilated in order to make a good repair decision. This knowledge includes:

(1) experience-based knowledge about repairs and ship condition,
(2) large volume of historical information from past ship inspections and repairs,
(3) complex ship structure information,
(4) complex loading information,
(5) complex analysis procedures, and
(6) expert knowledge of structural design, fracture mechanics and corrosion.

Poor or incomplete repair decisions are often made simply because there is not enough time or money to perform a detailed analysis. It is apparently that a tool needs to be developed for the management of the information used to make rational repair decisions.

This poses the key question addressed in this research: **How do we properly manage the computerized inspection and repair data, the existing knowledge of both successful and unsuccessful repairs, the complex analysis tools and additional knowledge to make intelligent and timely repair decisions?** The answer proposed by this research is the **Repair Management System (RMS)**. The RMS is a computerized framework to help repair engineers make good repair decisions by assisting engineers with structural failure diagnosis and repair alternative evaluation. The RMS is the first known attempt to handle the complexities of ship structural repair analysis in a framework that provides both elements critical to good repair--quick decisions and thorough evaluations.

The objectives of the RMS project were to:

(1) develop a framework for the development RMS,
(2) develop the second version of the software RMS for more ship critical structural details, and

Rank (1=most desirable feature)

Feature	A	B	C	D	E	F	G	H	Avg.
Expected life analysis of repair alternatives	1	5	3	1	1	1	2	3	2.1
Economic tradeoff analysis of repair alternatives	4	6	5	5	3	2	3	1	3.6
Graphical database of possible repairs	2	4	1	3	2	4	1	2	2.4
Extendibility to allow updating with new repair data	5	2	4	4	6	3	5	6	4.4
Repair database analysis capabilities (statistical)	3	3	6	6	5	5	4	4	4.5
Reliability-based information	6	1	2	2	4	6	6	5	4.0

Table 1.1. Results of Repair PC Code Questionnaire

Since some important features were not included in the first version of RMS due to the limited time and the RMS promised a potential to become a powerful tool, a new research project was proposed and approved. The **RMS 2.0** was developed and this report documents the results.

1.2 Ship Maintenance and Repair

After a new ship is delivered, the ship's hull structure must be monitored by a series of internal and external inspections to assess the integrity of the ship structure. The scope and frequency of these inspections are determined by classification society, owner/operator or U.S. Coast Guard guidelines. These inspections provide means to evaluate the current condition of steel and coatings and to detect unexpected flaws and damages, and permit appropriate maintenance and repair measures to be taken to preserve the integrity of the hull structure. During an inspection, several types of structural failures

CHAPTER 2. BASICS OF SHIP MAINTENANCE

This chapter provides a general introduction to ship maintenance and repair in details. Ship structural maintenance involves three levels: inspection, maintenance and repair. Inspections are to uncover structural problems such as cracks, buckling, corrosion, pitting/grooving and coating breakdown. Preventative maintenance is to address problems before they occur like using coating to prevent steel from corrosion. Repair of structural problems follows discovery by inspection. Since this chapter doesn't cover the materials of RMS but serves to introduce the basics of ship maintenance, those who are familiar with ship maintenance may want to skip this chapter.

2.1 Inspections

After a new ship is delivered, the ship's hull structure must be monitored by a series of internal and external inspections to assess the integrity of the ship structure. These inspections provide means to evaluate the current condition of steel and coatings and to detect unexpected flaws and damages, and permit appropriate maintenance and repair measures to be taken to preserve the integrity of the hull structure.

Before an inspection, appropriate planning and preparation are important. The purpose and scope of the inspection should be identified first. The scope of the inspection is depended on the inspection program. For each inspection, the extent of areas to be inspected should be specified. Generally, four basic defects will be recorded during all types of inspection. They are cracking, corrosion, coating breakdown and buckling. Additionally the inspector assesses the following conditions: corrosion rates, pitting, percentage of pitting covering the plate, piping and fittings, handrails, ladders and walkways.

The scope of internal structural inspections as required by the Classification Societies is listed in the following Table 2.1 [TSCF, 1986]. In this table, it can be seen that the extent of the requirement increases with the age of the ship. An overall survey is a survey intended to report on the overall condition of the tanks' structural integrity and corrosion condition in a relatively short period of time and to determine the extent of additional close-up surveys requirements. A close-up survey is one where the structural components are within the inspection range (within arm's reach) of the surveyor. In practice, the areas that will be inspected first will be those that are most accessible. However, as the age of the ship increases, additional access for close-up inspection will be necessary for most areas of the structure. This close-up survey is particularly necessary for crack detection, corrosion assessment and thickness measurement.

The minimum requirements for thickness measurements is listed in Table 2.2 [TSCF, 1986]. The number of locations and extent of surveys are greater in the permanent ballast tanks and in tanks used primarily for water ballast because these tanks are subjected to a more corrosive environment. In addition to the thickness measurement specified in precise locations, sufficient measurements are required to assess and record corrosion patterns.

Since the size of ship structure is enormous, it is almost impossible to perform a 100% inspection. The inspectors must have a good understanding of the structural layout and crack history of this ship. Information should be obtained prior to the commencing of the survey. This includes structural drawings, previous inspection data, previous repair records, condition and extent of protective coatings, operational history, and so on. Combining this information with the inspectors' experience, they can determine where to inspect more efficiently. In addition, inspectors need to know the locations of critical structural details with high likelihood of failure. Discussion with all involved parties, including the ship's staff, classification society, and ship representatives, can give inspectors insight into the locations of critical areas. If an inspection database is available, it will give inspectors further insight into where and when to expect structural damage and defects. Areas that are of concern to the inspector with respect to fracture initiation are listed below [TSCF, 1986]:

- Ends of principal girders, stringers, transverses and struts with associated brackets. Particular attention should be paid to toes of brackets.
- Bracketed ends to shell , deck and bulkhead stiffeners.
- Connection of shell, deck and bulkhead longitudinals to transverse web frames and bulkheads. Particular attention should be paid to the side shell connections between full load and ballast waterlines.
- Any discontinuity in the form of misalignment or abrupt changes of section.
- Plating in way of cutouts and openings.

- Areas that show any evidence of damage or buckling.
- Erection butts in plating and longitudinal stiffeners.

For corrosion concern, the bottom is perhaps the most commonly inspected area in a tanker. The extent of wastage should be checked. For coated tanks, wastage will take the form of localized pitting and grooving in way of coating failure. Generally, inspections for localized corrosion can be focused in the following areas:

- Top and bottom of ballast tanks,
- Bottom of cargo tanks where pitting corrosion could occurs,
- Any horizontal surface which can entrap water, in particular, horizontal stringers on transverse bulkheads,
- Welds, sharp edges, and any areas in which coating is difficult to apply,
- Local stiffening members which can become the sites of grooving corrosion, and
- Zinc Anodes.

A good way to keep track of the trend of critical areas is to use a computerized database system. A computerized database system is used for typical defect documentation and inspection results. It can simplify the handling of gauging and inspection data. Besides, developing high quality databases on corrosion and cracking histories and containing sufficient volumes of data can assist in defining the areas of the hull structure that should be closely inspected and monitored on a more frequent basis.

Table 2.1: Inspection program [TSCF, 1986]

Age < 5 years Special Survey No. 1	5 < Age < 10 Special Survey No. 2	10 < Age < 15 Special Survey No. 3	15 < Age < 20 Special Survey No.4
1. Overall Survey of all tanks and spaces 2. Close-up Survey: a) One complete transverse web frame ring including adjacent structural members (in one ballast tank if any, or a cargo tank used primarily for water ballast) b) One deck transverse including adjacent deck structural members in one cargo wing tank c) Lower part of the girder system including adjacent structural members on one transverse bulkhead in one ballast tank, one cargo wing tank and one cargo center tank	1. Overall Survey of all tanks and spaces 2. Close-up Survey: a) One complete transverse web frame ring including adjacent structural members in one wing (in one ballast tank if any, or a cargo tank used primarily for water ballast) b) One deck transverse including adjacent deck structural members in each of the remaining ballast tank, if any c) One deck transverse including adjacent deck structure in one cargo wing tank and two cargo center tanks d) The complete girder system including adjacent structural members on the transverse bulkheads in one wing tank (in one ballast tank, if any, or a cargo tank used primarily for water ballast) e) Lower part of the girder system including adjacent structural members on one transverse bulkhead in each of the remaining ballast tanks, one cargo wing tank and two cargo center tank	1. Overall Survey of all tanks and spaces 2. Close-up Survey: a) All complete transverse web frame rings including adjacent structural members in all ballast tank and in one cargo wing tank b) One complete transverse web frame ring including adjacent structural members in each remaining cargo wing tanks and one bottom and one deck transverse in each cargo center tank c) One complete girder system including adjacent structural members on the transverse bulkheads in all cargo and ballast tanks	1. Overall Survey of all tanks and spaces 2. Close-up Survey: as for Special Survey No. 3 with additional transverses as deemed necessary by the Surveyor

Table 2.2: Minimum requirements of thickness measurements at special hull surveys of oil tankers [TSCF, 1986]

Age < 5 years Special Survey No. 1	5 < Age < 10 Special Survey No. 2	10 < Age < 15 Special Survey No. 3	15 < Age < 20 Special Survey No.4
1. One section of deck plating for the full beam of the ship within 0.5 L amidships (in way of a ballast tank, if any, or a cargo tank used primarily for water ballast) 2. Sufficient measurements of structural members subject to Close-up Survey for general assessment and recording of corrosion pattern 3. Suspect areas	1. Within 0.5 L amidships: a) Each deck plate b) One transverse section 2. Sufficient measurements of the different structural members subject to Close-up Survey for general assessment and recording of corrosion pattern 3. Suspect areas 4. Selected wind and water strakes outside 0.5 L amidships	1. Within 0.5 L amidships: a) Each deck plate b) Two transverse sections 2. Sufficient measures of the different structural members subject to Close-up Survey for general assessment and recording of corrosion pattern 3. Suspect Areas 4. Selected wind and water strakes outside 0.5 L amidships	1. Within 0.5 L amidships: a) Each deck plate b) Three transverse sections c) Each bottom plate 2. Sufficient measurements of the different structural members subject to Close-up Survey for general assessment and recording of corrosion pattern 3. Suspect areas 4. Selected wind and water strakes outside 0.5 L amidships

Table 2.3: Minimum requirements of tank testing at special hull surveys of oil tankers [TSCF, 1986]

Age < 5 years Special Survey No. 1	5 < Age < 10 Special Survey No. 2	10 < Age < 15 Special Survey No. 3	15 < Age < 20 Special Survey No.4
1. Cargo tank boundaries facing ballast tanks, void spaces, pipe tunnels, fuel oil tanks, pump rooms or cofferdams	1. Cargo tank boundaries facing ballast tanks, void spaces, pipe tunnels, fuel oil tanks, pump rooms or cofferdams 2. All cargo tank bulkheads which form the boundaries of segregated cargoes	1. Cargo tank boundaries facing ballast tanks, void spaces, pipe tunnels, fuel oil tanks, pump rooms or cofferdams 2. All remaining cargo tank bulkheads	Cargo tank boundaries facing ballast tanks, void spaces, pipe tunnels, fuel oil tanks, pump rooms or cofferdams 2. All remaining cargo tank bulkheads

2.2 Maintenance

The most critical structural problem found on aging vessels having suffered from lack of long term preventive maintenance is severe corrosion of hull structures, particularly in permanent ballast tanks. Such tanks are normally provided with coating at the new building stage. If not properly maintained, this coating will normally break down and lose its preventive effects after 5 to 10 years. Thereafter an increased rate of corrosion will be experienced. At the time when such vessels come up for their third special periodical survey (12-15 years of age) it will normally be necessary to renew significant amounts of steel mainly in the form of internal structures. To prevent expensive steel renewing, coating should be maintained constantly.

By means of maintaining the coating well, the hull structure may last for 25 years and beyond without the need for steel renewals, even in permanent ballast tanks. On the other hand without maintaining the corrosion protection system, the need for significant steel renewals will normally start at around 15 years of age [DnV, 1991]. Since steel renewals are expensive, the coating repair is critical for owners. By deferring coating repairs, the owner risks steel renewals at the next overhaul. Roughly speaking, the cost to coat plating is equal to the cost of renewing 10% of the same plate assuming a thickness of 12 mm [Tikka, 1991]. Besides, steel work in an existing structure introduces new problems such as residual stresses and possible weld defects. Thus, if corrosion has result in critical coating breakdown, such tanks are recommended to be blasted and re-coated timely.

From both visual and gauging information of a survey, decisions can be taken regarding life continuance and to the extent of maintenance necessary to reinstate the corrosion protection system. In the case of long-term (8 to 10 years) operations, re-coating of the breakdown areas (or more usually the entire tank) would be regarded as a cost effective solution instead of any potential steel renewals. For shorter-term (4 to 5 years) operations, temporary protection systems such as soft coatings or sacrificial anodes may be considered. The effective life of soft coatings is usually restricted to about 2 to 4 years only, for this reason this protection system should really be regarded as temporary and should be subjected to more regular and comprehensive thickness gauging and close-up surveys than that considered for hard coatings [TSCF, 1992].

2.3 Repairs

The repair of critical internal structural details is a difficult and demanding task for ship owners. There is no reasonable consensus on what, how, and when to repair. The general lack of readily retrievable and analyzable information on repairs and maintenance frustrates repair and maintenance tracking. Take crack repair for an example. Many crack repairs appear to be ineffectual. Veeing and welding cracks that have occurred early in the life of the ship seems to be ineffective; they quickly develop again. If one replaces the cracked plate and modify design by adding a bracket, a lug, or etc., the repair can usually last longer than veeing and welding. However, this repair may not be cost effective if the ship will be scrapped in the near future.

Three types of repairs, crack repair, steel renewal and pitting/grooving repair, are introduced in the following paragraphs.

2.3.1 Crack Repair

Cracks are potentially the most serious of defects as they can grow rapidly in size leaving affected structure unable to bear loads. As a result, the surrounding structure must carry a greater loading that can in turn lead to its failure in the future. If this process continues unchecked, hull girder or long large panels of side shell collapse can result.

Repair of cracks vary widely. Repairs of cracks can range from temporary cold patches to stop leaks to complete re-design of the structural detail and replacement of steel nearby the detail. Welding cracks is a popular repair, but it frequently failed again within a short time. Drilling the ends of the cracks is a frequently used temporary repair measure that is used until the ship can be taken into the dry-dock. Repairs of these cracks can range from simple welding to addition of reinforcing elements. Experience indicates that many of these repairs must be repeated in subsequent dry docking. In one case, a series of side shell longitudinal crack has been repaired four times, and each time a different repair procedure has been tried [Bea, 1992].

been projected for the vessel. For instance, in the case of the removal and re-welding of bulkhead stiffening to bulkhead plating, the chances of penetrations of the remaining corroded plating is usually very high and the future watertight integrity of this division remains in-question. Also, the combination of steel renewal and coating could be the most cost effective method for a longer life span.

Table 2.4: Criteria of wastage for local strength of structural components [TSCF, 1986]

STRUCTURAL COMPONENT	% CORROSION (1) LOSS INDICATOR		BUCKLING GUIDELINES (LONGITUDINAL FRAMING)	
	A(2)	B(3)	Mild Steel	HTS 36
Deck and bottom plating and longitudinal girders	10	25	s/t = 55 to 60	s/t = 49 to 52
Webs of deck and bottom longitudinals	15	30	h/t = 50 to 65	h/t = 45 to 55
Flat bar longitudinal at deck and bottom (4)	10	25	h/t = 15 to 20	h/t = 15 to 17
Face plates and flanges of longitudinals and longitudinal girders	15	25	b/t = 10	b/t = 10
Side shell	-	20	(5)	
Longitudinal bulkhead plating	15	25	s/t = 70 to 75	s/t = 60 to 79
Webs of side shell and longitudinal bulkhead longitudinals	-	25	(5)	(5)
Transverse bulkhead structure, transverses and side stringers	15	25	(6)	(6)
Remaining secondary structure	-	30	-	-

Notes
(1) Percentages are to be applied to original Rule thicknesses without corrosion allowance reductions for corrosion control notation.
(2) Column A refers to percent reductions above which further assessment is required.
(3) Column B refers to percentage reductions where steel renewals may be required.
(4) The deck and bottom plating and associated longitudinals are to include side and longitudinal bulkhead plating and associated longitudinals within 10% of the depth of ship from the deck and bottom respectively.
(5) No buckling guidelines are given as the components are not usually limited by this.
(6) Due to the wide variation in stress levels and stiffening arrangements, no general guidance figure can be given. Individual guidance should be sought from the Classification Society concerned.

Definitions
t = thickness of structure after corrosion.
s = spacing between longitudinal stiffeners.
h = web depth of longitudinal stiffeners.
b = half-breadth of flange for symmetrical sections, and the flange breadth for asymmetrical sections.

In some cases generally corroded areas of tank structure are found to be below the minimum section modulus requirements. It may be possible, at the discretion of the relevant Classification Society, to install additional steelworks in conjunction with an effective corrosion protection system (painting), rather than carry out extensive steel renewals. This form of repair should aim at re-establishing the required minimum section modulus of the overall defective areas, while dealing directly with local defects or fractures as found necessary. Regular re-inspection of this alternative reinforcement should be carried out to ensure its continued effectiveness in maintaining the overall structure integrity of the vessel [TSCF, 1992].

2.3.3 Pitting and Grooving

Pitting mainly can be found on the internal horizontal surface, particularly in the bottom plate of the cargo or ballast tanks. If widely scattered, they may not affect the general strength of the vessel. However due to their depth and quick deterioration rate, they may quickly lead to a through penetration with subsequent pollution danger. Using the corrosion rate of about 1 to 3 mm per year for pitting/grooving and the period to next overhaul, a **defined minimum thickness** can be established for the decision of pitting repair [Ma and Bea, 1992]. For examples, if the period to next overhaul is 5 years, the pits can grow about 15 mm deeper during these 5 years. To prevent pollution or water tight problems, the **defined minimum thickness** should be set as 15 mm at least in this case. Pitting repairs can be classified into three level according the remaining plate thickness. While the remaining plate thickness is more than the defined thickness, the pitting is recommended to be grit blasted and then brush coated with two coats of coal tar epoxy or to be vacuum blasted and filled with pourable pit filler. While the Remaining thickness is between the defined thickness and 6 mm, it is recommended weld up the pitting. If the pitting is so severe that the remaining thickness is less than 6 mm, it should be cropped and renewed with a new plate.

Grooving of structural members is another form of local corrosion which takes place usually next to weld connections and is related to flexing of the stiffened panel or areas of regular erosion. Epoxy coating of the affected areas and additional stiffening of the relevant panels is regarded as the best way of this problem.

CHAPTER 3. REPAIR MANAGEMENT SYSTEM

The purpose of this chapter is to review the inspection, maintenance and repair of ships. look at all the factors that go into an intelligent repair decision to demonstrate the complexity of the process. This chapter also discusses the approach used by the Repair Management System (RMS) to handle this complexity.

3.1 Repair Decision Steps

In any structural repair situation, there are four basic steps to determining the best repair. These steps are summarized below [Gallion, 1992].

3.1.1 Step 1 - Inspection on Structural Failure

Visual structural inspection on ships is performed at regular intervals to locate structural failures and describe the basic properties of the failures. These properties include crack location, crack orientation, crack length, percentage plate wastage and other information necessary to analyze the failure. Due to the enormous size, poor lighting, and dirtiness of the tanks, visual inspection is considered a "heroic" task that cannot locate all structural failures. The probability of crack detection governs the probability that a certain size crack will be detected during an inspection.

3.1.2 Step 2 - Determine Mode of Structural Failure

Various ways have been proposed to categorize modes of failure, including by loading type, stress type and others. The Ship Structures Committee categorizes cracks into two levels of crack severity [Stambaugh,1990]. Nuisance cracks are small cracks detected before they propagate into adjacent structure. Nuisance cracks are usually repaired by welding. Significant fractures are serious cracks that usually propagate perpendicular to the longitudinal and pose a serious threat to structural integrity, including a loss of watertight integrity or complete failure. For this research, both nuisance cracks and significant fractures are arranged into two load categories of ship structural failure--dynamic and static loading failure.

Selecting crack repair method can depend on the location of the crack. Cracks in primary structure require more serious repair than those in secondary structure. Primary structure is the structure which contributes significantly to the main structural strength of the ship such as hull plates, stiffeners, principal decks, main transverses, and so on. Secondary structure is the structure which neither contributes to the structural strength nor the watertight integrity such as partition bulkheads, platforms and so on.

Cracks in primary structure may be temporarily repaired by fitting double plates or gouging out the crack and filling in with weld metal. Gouging and re-welding is an easy and common way of repair. However, the strength of re-welding cracks is, almost invariably, worse than the original one. The repaired weld will create new crack potentials and thus fail even earlier. Such repairs are sometimes considered in attempting to get the ship to a facility where full repairs can be made. The better and formal ways of repair are to crop and renew the cracked plate or to modify the local geometry to reduce the stress concentration. If a longer life continuance is expected for the ship, a more robust repair such as geometry modification should be considered.

In the other hands, cracks in secondary structure may be arrested temporarily by drilling a hole of diameter equal to the plate thickness at a distance of two plate thicknesses in front of the visible crack tip and on a line with the direction of anticipated crack propagation [Ma, 1992].

It is difficult to decide which repair method is most reliable and cost effective for a particular crack. The selection of different repair alternatives is usually depended on the location of the crack and the expected life continuance of the ship.

2.3.2 Steel Renewal due to Corrosion or Buckling

In the event of steel renewals being required to compensate for either local corrosion wastage or buckling, according to the following acceptance criteria in Table 2.1, it is important that the extent of this new material is sufficient to maintain structural continuity and avoid any potential discontinuities. From the repair point of view, the replacement of complete panels of structure may prove most cost effective and ultimately more reliable, than merely renewing individual members especially if a longer life span has

3.2 RMS System

For the RMS, knowledge can take heuristic (rule-based), probabilistic and numerical forms. These forms include: (1) heuristic/probabilistic knowledge about mode and cause of failure; (2) heuristic knowledge about valid repair alternatives; (3) numerical routines for alternative evaluation; and (4) heuristic or probabilistic decision analysis. Since this knowledge is not simply heuristic, the RMS is a "coupled" expert system that requires both symbolic and numeric processing. The type of information required to evaluate these steps is summarized in the following Table.

Step	Description	Computational Requirements
1	Gather Data	Data
2	Determine Mode of Failure	Knowledge
3	Determine Cause of Failure	Knowledge
4	a. Determine Repair Alternatives	Data+Knowledge
	b. Evaluate Repair Alternatives	Data+Knowledge+Numerical
	c. Select Repair Alternative	Knowledge

Table 3.1: RMS Computational Requirements [Gallion and Bea, 1992]

The overall architecture of an ideal RMS would consist of the user interface, knowledge-base, database, analysis procedures and inference engine--as detailed in Figure 3.1. To organize the wide array of knowledge required for repair analysis, the knowledge in the RMS is grouped together into several module, each of which require different knowledge representation schemes. These modules include the following:

- control module;
- failure diagnosis module;
- repair alternatives selection module;
- repair analysis module; and
- decision analysis module.

longitudinal near the tip of a beam bracket (see Figure 3.3). This type comprises 32.8% of total cracks. These two types of cracks totally consists of 45.1% of cracks.

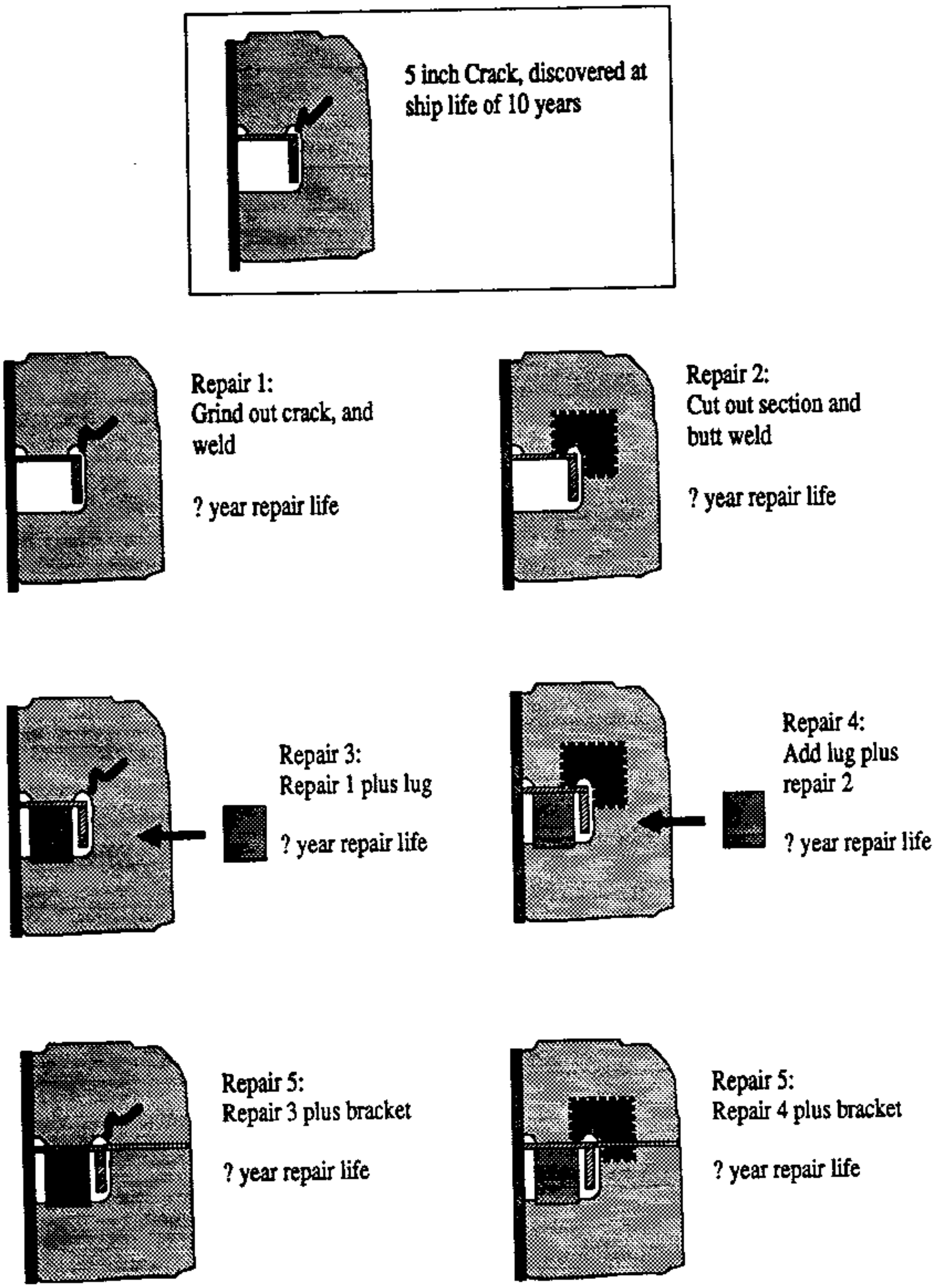


Figure 3.2: Repair alternatives example 1 [Ma, 1992]

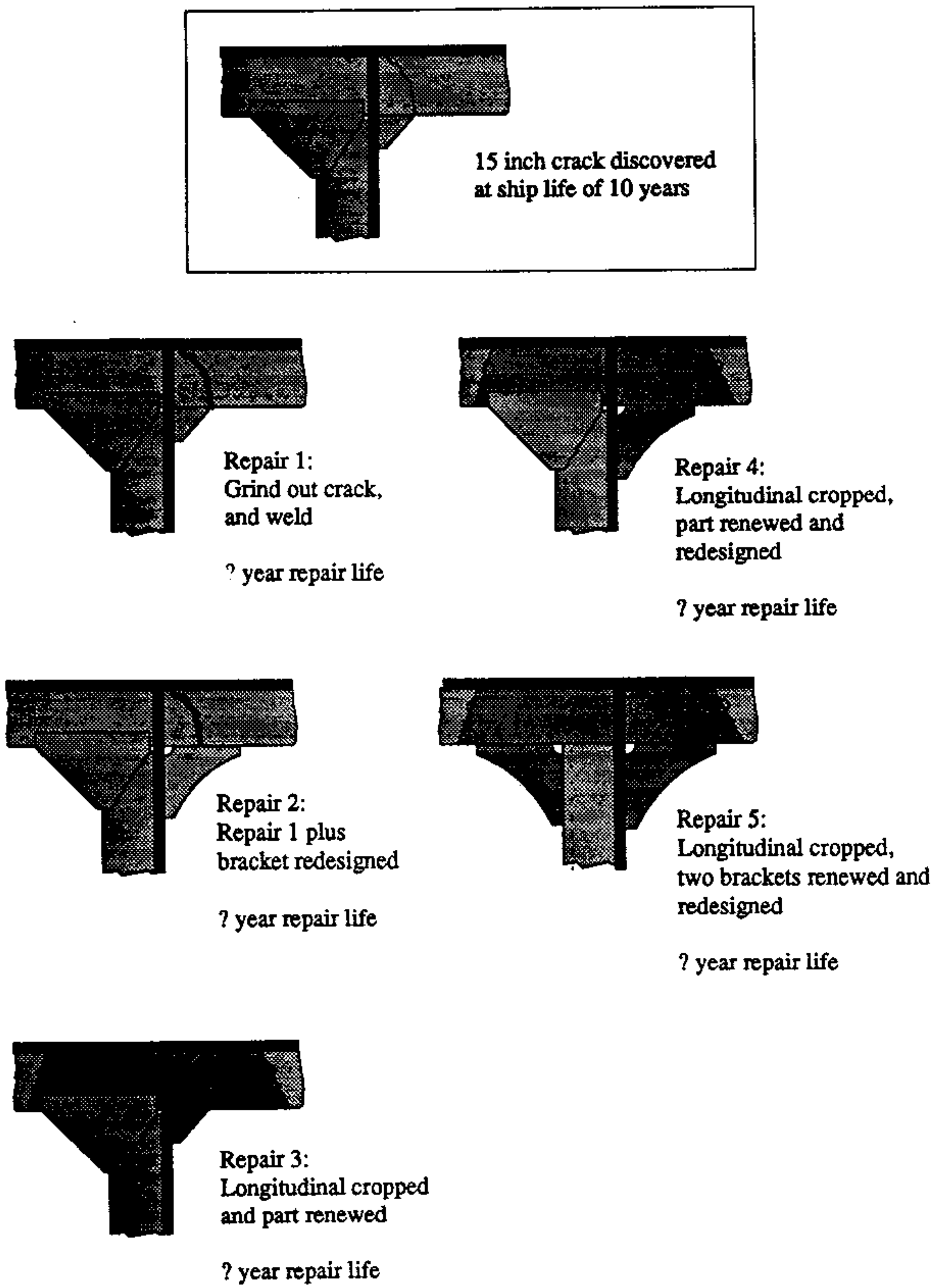


Figure 3.3: Repair alternatives example 2

3.2.4 Numerical Analysis Modules

Analysis is conducted by the analysis modules. The type of analysis required is determined by the results of the failure diagnosis. For example, if the failure mode is high cycle fatigue with a high degree of certainty, then a fatigue analysis would be required. Various types of analyses might be required, including:

- fatigue analysis;
- corrosion analysis;
- buckling analysis;
- global failure analysis; and
- structural reliability and condition assessment analysis.

These modules serve to link symbolic information concerning analysis steps, numerical procedures and interpretation of numerical results to conduct analysis.

Since ship repair engineers are often unfamiliar with the details of fatigue, fracture, corrosion, and other analyses as applied to the complex case of a ship structure, the modules associated with these analyses could also serve to educate the users through an extensive explanation facility. To account for the different structural configurations, a library of standard structural details is required in the general database. New details must be added as required.

A probabilistic approach to the calculations in which the historical database is used to establish a prior probability of failure for a particular structural detail could be incorporated into these modules.

3.2.5 Decision Analysis Module

A final module, the decision analysis module, is required to select the most appropriate repair alternative. A structured procedure is required due to the high level of uncertainty involved in the various stages of the analysis. These uncertainties are associated mainly with the following:

- Environmental Factors. The primary environmental factor is corrosion of the ship structure due to inadequate maintenance.
- Combined Effects.

In reality, structural failures usually result from combined effects. Two or more factors usually contribute to the cause of damage in varying degrees. For example, the environmental factor of corrosion exists in some form for most ship structural failures but is not always the primary cause of damage. The Ship Structural Committee has categorized the causes of fracture in a similar manner. These categories include abnormal forces, presence of flaws or notches, inadequate physical properties at service temperature, and combination of causes [Stambaugh,1990].

3.1.4 Step 4 - Evaluate Repair Alternatives and Select

Once the mode and cause of failure have been determined with a degree of certainty, alternative repairs can be evaluated. This step is one of the most difficult due to the large number of factors that should be considered. The repair that best satisfies the life continuance, economic, location, time and other considerations is the one that should be chosen. These repair considerations are discussed in the following section.

Life continuance consideration can be the most important factor in repair decisions. For example if a ship is going to be kept in service for another 5 years and then retires or to be sold, the ship owner may select a repair that can last for more than 5 years. Supposing the repair work well, the failed critical structural detail will be out of trouble for the rest of 5 years with a high reliability. This consideration is related to the economic consideration. However the difficult part is the life estimation of a particular repair method. It will be investigated later in this report.

Economic considerations can play a dominate role in repair decisions. These economic factors include the future plans for the ship, age of the ship, total cost and time to complete repairs, cargo transport obligations, money available, current steel costs, repair rates, wage rates, etc. The economic decision is usually based on the certain initial repair costs and not the possible future costs of maintenance. This is mainly due to the complexity of the repair decision, which makes future costs difficult to evaluate. However, future costs for inadequate, non-durable repairs may dominate the decision. A complete economic analysis should take into account the tradeoff between initial and future costs. In the same way that a more durable ship has lower maintenance costs, more durable repairs will have lower future repair costs.

Repair location consideration falls into two categories. Voyage repairs are made at sea mostly in emergency situations. Voyage repairs are often very difficult since "hot work" (welding) is usually prohibited in critical hull structure due to the presence of flammable materials. As a result, cold patching is a popular temporary remedy. Shipyard repairs are made either at dockside or in a dry-dock environment after the tanks are ventilated and washed to accommodate hot work in the tanks. This is the most ideal repair environment although it still presents problems due to the enormous size of crude oil carriers.

Time considerations include factors such as the time available to complete repairs and the time until the next inspection and repairs. More thorough repairs are required if there is a long time before the next inspection or overhaul period.

Several additional considerations must be taken into account in repair alternative evaluations. These considerations include the following: Classification societies like American Bureau of Shipping (ABS), Bureau Veritas, Det Norske Veritas, Germanischer Lloyd, Lloyd's Register of Shipping and others dictate the minimum structural requirements for compliance with class rules. Also Regulating authorities, such as the United States Coast Guard, dictate the minimum requirements for ship operation within their jurisdiction. Environmental safety has become a major consideration in the repair of ships. Environmental disasters can produce both ecological damage and serious financial damage to the owner and operators of the ship as illustrated by the grounding of the Exxon Valdez in Prince William Sound [Davidson, 1990]. The goal of repairs is to minimize the chance that such an incident is caused by poor repair and maintenance of the structure. Accessibility for monitoring by crew will determine whether monitoring of minor structural problems is feasible. If a structural failure cannot be monitored effectively it must be repaired.

The **dynamic failure mode** occurs under the condition of cyclic loading and includes the following specific modes of failure:

- Low cycle fatigue failure occurs under cyclic loading of 0.5 to 1000 cycles. Loads generally exceed the yield strength of the material. Failure occurs by rapid crack initiation and growth.
- High cycle fatigue failure occurs under cyclic loading of 1000 cycles or more. The endurance limit of a material ("infinite" life) exists when failure cannot occur below a certain stress level. Failure is predicted by the Goodman diagram approach or by Linear Elastic Fracture Mechanics (LEFM) techniques using the Paris equation. Failure occurs by crack initiation and growth. Cracks already exist in welded structure in the form of weld imperfections and failure occurs by crack growth only. The fracture surface is usually flat and contains small lines (beach marks) that radiate out from the crack origin .
- Corrosion fatigue is the acceleration of crack propagation in the presence of cyclic loads in a corrosive environment, such as sea water.

The **static failure mode** occurs under the condition of static loading and includes the following specific modes of failure:

- Brittle fracture occurs under static loading and is typical in materials with yield strengths less than 0.5 percent strain before fracture, such as cast iron, concrete and ceramic. Failure is predicted fairly accurately by the maximum normal stress theory and occurs by fracture (not yielding). Materials that are not normally brittle can become brittle in some environments, such as low temperatures. The fracture surface is usually flat and contains arrow shaped lines known as "Chevron marks" which point to the origin of the failure.
- Ductile fracture occurs under static loading and is typical in materials with yield strengths greater than 0.5 percent strain before fracture, such as steel and aluminum. Failure is predicted by several failure theories, including the maximum shear stress theory and the distortion energy theory (von Mises). The fracture surface is usually distorted due to failure by yielding.
- Buckling failure occurs under compressive loading under sufficient load to surpass unstable equilibrium. Standard solutions exist for bucking of a simple column under compression with various end constraints. More complicated structure, such as the plate structure of a ship, is a difficult analytical problem that requires finite element techniques.

- Stress corrosion cracking can occur in parts subjected to continuous static loads in a corrosive environment. The degradation of strength is represented by the reduction of fracture toughness with time.

All the above modes are influenced by environmental factors. For example, general corrosion reduces plate thickness and increases both the static and dynamic stresses on the plate, possibly leading to a dynamic or static failure mode. As another example, hydrogen embrittlement would accelerate the advent of brittle fracture. In addition, a single fracture can contain several modes. For example, a small crack that exists at a welding imperfection will grow in a stable manner by fatigue. At some crack length, the stress may reach a critical level and cause unstable crack growth by brittle fracture. This brittle fracture may be arrested by load sharing with adjacent structure or an increase in material thickness along the crack front.

Since a majority of ship structural failures are initiated by high cycle fatigue and corrosion effects, the RMS will concentrate in these areas. However, it is important to keep in mind these other possible modes. The mode of failure dictates the analysis procedures required to evaluate a failure.

3.1.3 Step 3 - Determine Cause of Structural Failure

There are five basic causes of a ship structural failure. These causes are the following:

- Design Problem. This cause includes insufficient static, fatigue and/or buckling strength in the design. This insufficiency could result from poor analysis procedures, poor material selection for the service conditions, underestimation of loadings and/or incorrect or insufficient structural modeling.
- Insufficient Quality Control. This cause occurs during construction and results in faulty material processing or fabrication. Examples include poor or incorrect welding procedures, incomplete welding, material defects and tolerance problems.
- Overloading. This cause includes situations that cannot be foreseen in initial design. Examples include collisions, poor tug operations and poor seamanship in extreme weather.

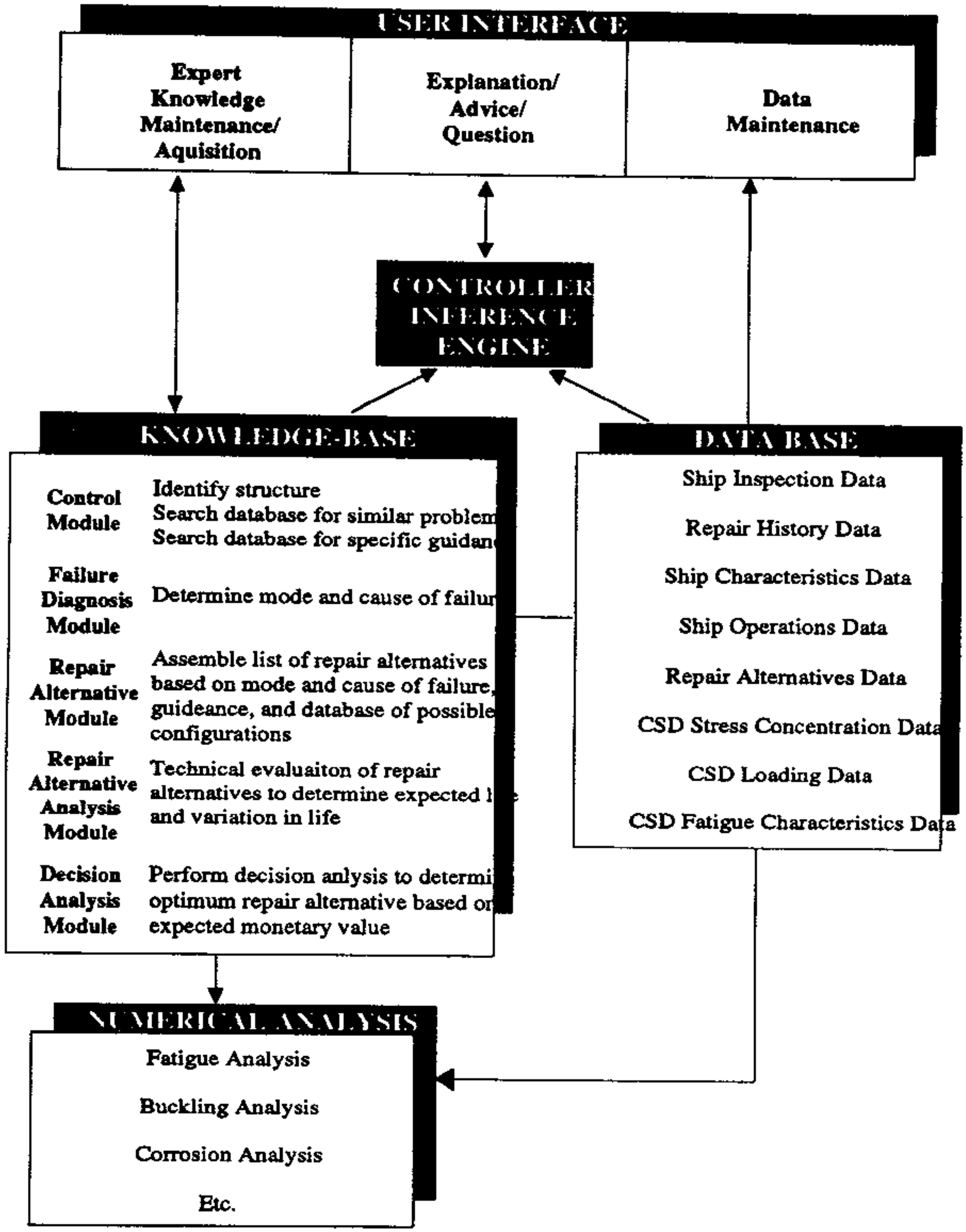


Figure 3.1: RMS System Architecture [Gallion and Bea, 1992]

- mode and cause of failure;
- repair life analysis procedure;
- cost estimates; and
- economic variables.

Depending on the repair option selected, the expected life of the repair and the uncertainty in life will vary. By accounting for the various economic factors and the uncertainties in the life estimation process, this module could help a repair engineer evaluate alternatives based on both initial and expected future costs, including the cost of failure.

3.2.1 Control Module

The control module is a guide to lead the user through the initial steps of making a repair decision. These steps include:

1. inspect the ship and input structural problems to database;
2. identify specific structural detail and failure to evaluate;
3. search ship condition database to determine if similar problems encountered and if past repairs successful or unsuccessful; and
4. search repair guidance database for specific information about structural problems.

This module would combine heuristics with database search procedures.

3.2.2 Failure Diagnosis Module

The failure diagnosis module would be a guide to evaluate the mode and cause of the structural failure based on the physical appearance of the failure, location of the initial failure, the orientation of the failure, the location in the ship, the type of structural detail, and other factors. The result of this module would be a list of possible modes and causes with their associated levels of certainty.

This could include heuristic or probabilistic knowledge based on the opinions of experts in the field of ship structural mechanics and the ship condition and repair guidance database information.

3.2.3 Repair Alternatives Selection Module

The Repair alternatives selection module serves to select the viable repair alternatives based on the mode and cause of failure, the detail configuration and other considerations. Two repair types are developed. The first one is a crack in a longitudinal cutout (see Figure 3.2). According past studies [Jordon, 1978][Jordon, 1980], this type of crack comprises 12.3% of total cracks in some ships. The second one is a crack on a

CHAPTER 4. REPAIR ALTERNATIVE SELECTION

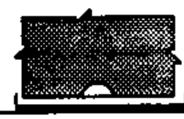
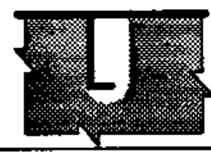
A ship structure may be classified into categories, ranging from global to detailed structure. The global hull can be simplified as a beam. To ensure this beam has sufficient longitudinal strength, the midship section modulus must be examined carefully during the design stage. The local strength of the structural details must also be determined. Generally, if is not possible to complete analyze all of the structural details to determines either their capacity or their fatigue strength. As a result many ships have suffered different degrees of local fatigue cracks. These fatigue cracks usually concentrate in a few types of ship structural details. It is important to recognize which detail types are more critical than the others. In this chapter the categories of the details with high failure rates are introduced. Several repair alternatives for some detail types having high failure rates as reviewed. Repair decision making processes are reviewed.

4.1 Categorizing Critical Structural Details

Past studies [Jordon, 1978] [Jordon, 1980] have been conducted to provide data on the performance of structural details, and to identify what types of details crack most frequently. In these studies, structural detail failure data were collected and classified into 12 detail families to provide guidance in the selection of structural detail configurations (Table 4.1). Various merchant and naval vessels were surveyed including 13 tankers, 12 containers, 9 navals, 5 combination carriers, 5 general cargoes, 4 bulk carriers, and 2 others.

The results of the survey show that 2252 of the total 6856 damaged locations, or 32.8%, were found in beam bracket connections. Tripping brackets have the second highest failure percentage, 23.1% (Figure 4.1). Miscellaneous cutouts are the third highest, 12.4%. Clearance cutouts are the fourth highest, 12.3%. It is amazing that these 4 detail groups comprised more than 80% fatigue cracks. Since all of clearance cutouts and most of beam or tripping brackets are in the connection of longitudinals and transverses, we can conclude that the most critical area of a ship structural is in such connections.

Table 4.1: Detail classifications [3.3]

Type #	Name	Functional Provision	Typical Configuration
1	Beam Bracket	Increase strength of framing and stiffening members at their supports.	
2	Tripping Brackets	Laterally support framing and stiffening members.	
3	Non-Tight Collars	Provide a connection from webs of framing and stiffening members to the plating of supports that have cutouts at the members.	
4	Tight Collar	Same as 3 above except also cover the cutouts to prevent passage of fluid or objects through the cutout.	
5	Gunwale Connection	Join the strength deck stringer plate to the sheer strake.	
6	Knife Edge Crossing	Permits complimentary stiffening systems on opposite sides of plate	
7	Miscellaneous Cutouts	Provide a wide variety of holes for access, drainage, ease of fabrication, cable ways, pipes, stress relief, etc.	
8	Clearance Cutouts	Provide a hole in an intersecting member to allow another member to go through.	
9	Structural Deck Cuts	Allow passage through decks for access, tank cleaning, piping, cables, etc.	
10	Stanchion Ends	Transfer loads between stanchions and deck supporting members.	
11	Stiffener Ends	Connect an un-bracketed non-continuing stiffener to a supporting member.	
12	Panel Stiffeners	Stiffen plating and webs of girders. These are non-load carrying members.	

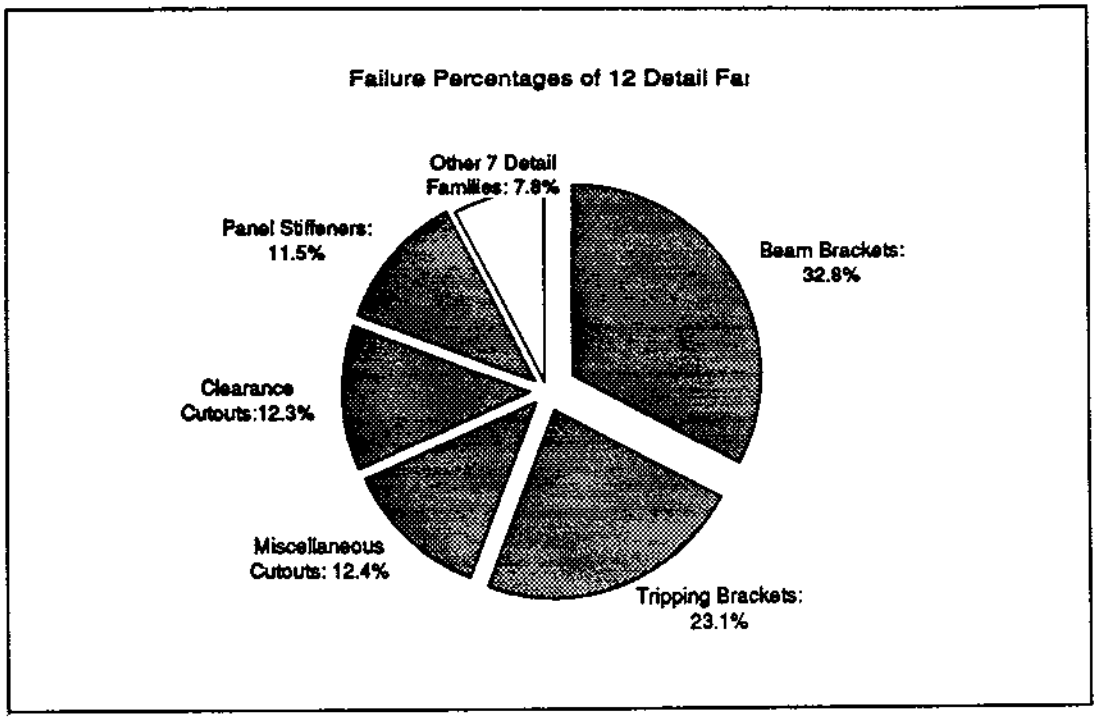


Figure 4.1: Failure percentage of 12 detail families

The transition from global structure to one example of this critical area in the ship side shell is illustrated in Figure 4.2. The fatigue crack database developed during the Ship Structural Maintenance Project indicates that side shell details experienced significantly more cracking than comparable details in bottoms, inner bottoms or decks [Schulte-Strathaus, 1991]. The main reason for this are the alternating sea wave loading that have stronger and more direct impact on the side shell plate between high water line and low water line.

50

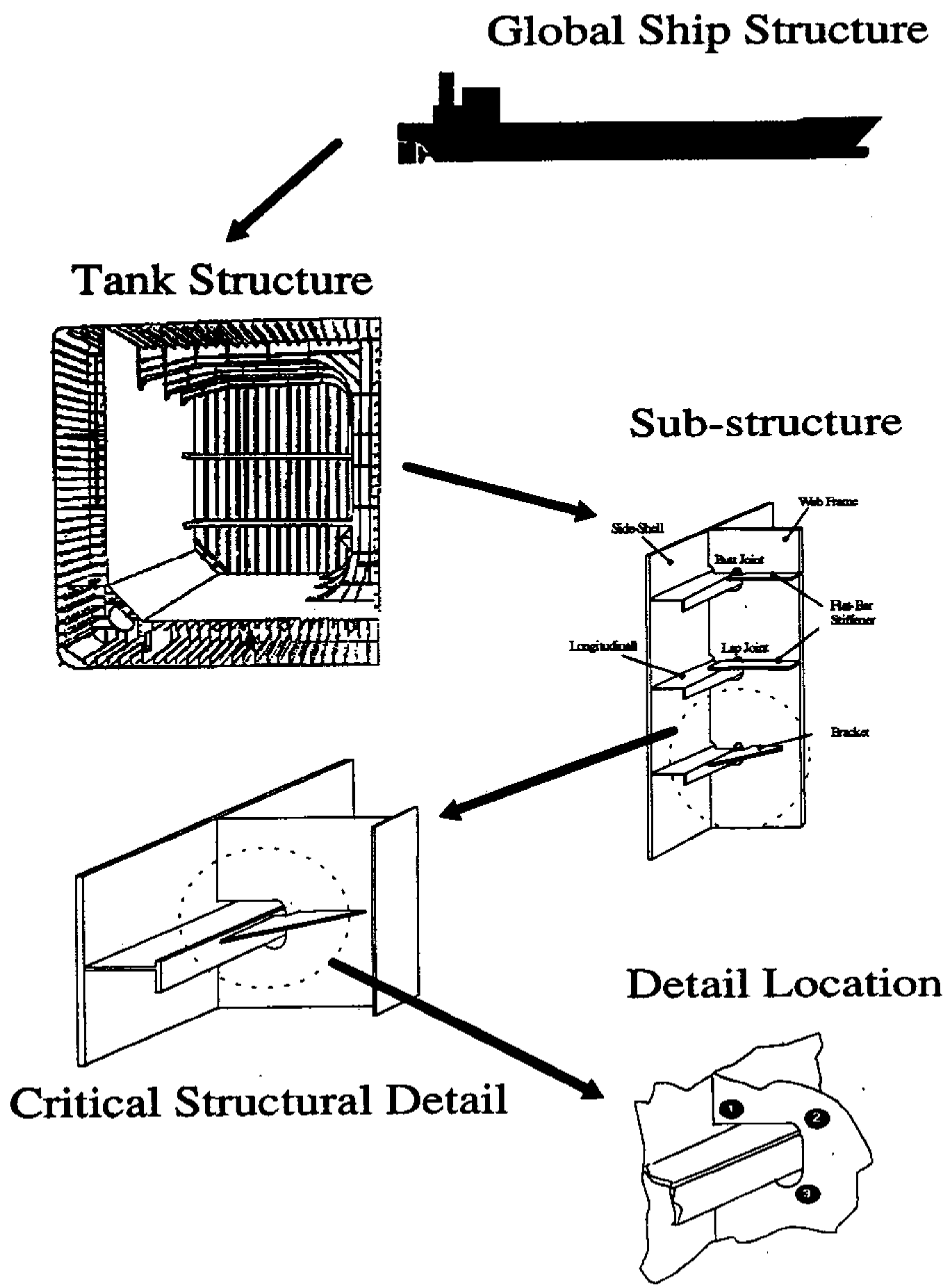


Figure 4.2. Global Structure to Side Shell Structure Components

4.2 Repairs Alternatives

For a fatigue crack in a particular structural detail, there are several methods to repair it. The expected repair life and repair cost of each repair method varies. Ship owners usually choose the most cost and time effective method. A robust but extremely expensive repair method may not be the best alternative. A less robust and cheaper repair may not be favored either, because later the repair may fail again. It will cost even more money to repair the detail again and again. Selecting a repair alternative requires a large measure of judgment and engineering insight.

The general strategies for crack repair of critical structural details can be classified as follows:

- **Drill a stopping hole in front of the crack tip (Temporary repair):** Cracks may be arrested temporarily by drilling a hole of diameter equal to the plate thickness at a distance of two plate thicknesses in front of the visible crack tip and on a line with the direction of anticipated crack propagation. Such repairs are sometimes considered in attempting to get the ship to a facility where full repairs can be made. It may also be used for cracks in secondary structure (the structure which neither contributes to the structural strength nor the watertight integrity such as partition bulkheads, platforms and so on).
- **Re-weld the cracks to the original construction:** Gouging and re-welding is an easy and common way of repair. However, the strength of re-welding cracks is, almost invariably, worse than the original one. The repaired weld will create new crack potentials and thus fail again in a shorter time interval.
- **Re-weld the cracks plus post weld improvement:** This repair is basically the same as the pervious one, except that the weld is ground into smooth to improve its fatigue strength. From the pervious study [Almar-Naess, 1985], the life extension effect of post weld improvement can be significant.
- **Replace the cracking plate:** This is also called inserting a new plate. The inserted new plate has a new clock counting of fatigue life. Since this plate has never carried any loads, its fatigue damage factor is zero. If the loading history and the material is identical to those of the failed plate, its fatigue life should be about the same as the failed time of the crack.

- **Modify design by adding bracket, stiffener, lug, or collar plate:** The more robust way of repair is to modify the local geometry to reduce the stress concentration. While adding a detail component and not involving cropping a large section, this repair may be one of the bests. It can reduce the stress concentration and therefore increase the repair life significantly. In addition it reasonably easy to apply.
- **Change configuration by applying soft toe, increasing radius, trimming face plate, enlarging drain holes, etc.:** This is another way to modify the local geometry to reduce the stress concentration. If a longer life continuance is expected for the ship, a more robust repair such as this should be considered.
- **Enhance scantling in size or thickness:** Increasing the size of a detail like a bracket is good. However increasing the thickness may not be a very good repair in the case that a discontinuity introduced to the plate. While doing this, the discontinuity should be carefully located outside the high stress area.

It is difficult to define which repair method is most reliable and cost effective for a particular crack. The selection of a particular repair alternatives depends on the location of the crack and the expected life of the ship. In the following paragraphs available repair alternative to three critical structural details with high failure rates are introduced.

4.2.1 Repairs of Beam Bracket

The beam bracket area is the most critical structural detail in ships according to past studies [Jordon, 1978] [Jordon, 1980]. A common failure in this area is the fatigue crack initiated from the bracket toe into the longitudinal (see the right side of Figure 4.3). Fatigue cracks can also develop along the connection line of the bracket and the longitudinal. Buckling through the middle of the brackets is another major failure mode in this area. Fatigue cracking in two typical configurations will be discussed to illustrate the available repair alternatives.

In the left side of Figure 4.3 is an example of bad design causing a cracking. Lacking a backing bracket induces a high stress concentration in the connection point. Six available repair alternatives are presented. Repair 4, 5, and 6 are the more durable repair choices since the added bracket can reduce the stress significantly.

In the right side of Figure 4.3 is another example of a fatigue cracking in the beam bracket area. Six available repair alternatives are presented. Still repair 4, 5, and 6 could be the better repair choices. Additional repair alternatives include using a soft (curved) bracket, soft nose and enhancing its size. Notice that backing brackets which are too small or do not incorporate a soft nose design may initiate fracture again from the bracket toe.

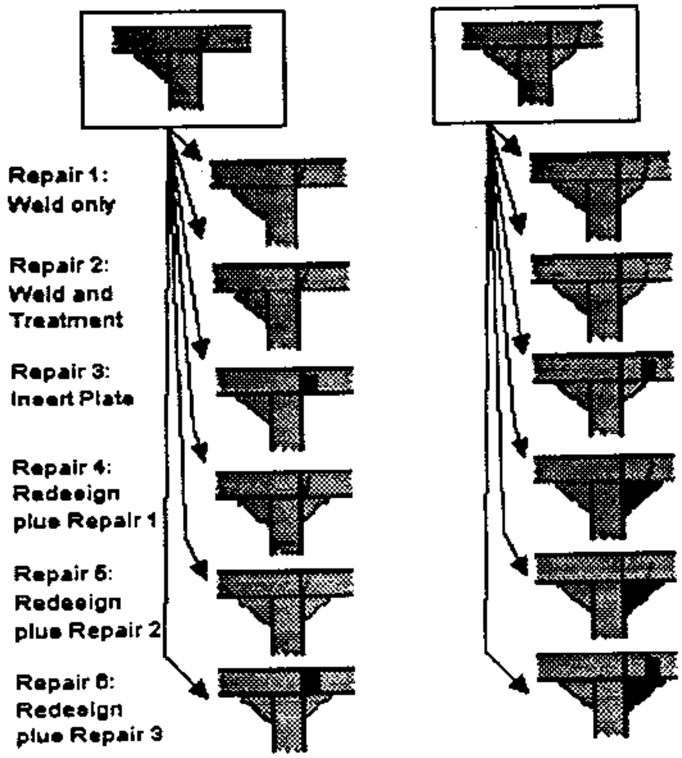


Figure 4.3: Repair alternatives for cracks in beam brackets

4.2.2 Repairs of Tripping Bracket

Tripping bracket is the second most critical structural detail. There are a variety of configurations of this type. The most common one is a flat bar stiffener on one side of the web. Other configurations include triangular or soft brackets on one or both side of the webs. Some brackets even have a flange. These details can be easily found in the connection of longitudinals and transverses. Three typical types of fatigue cracks are shown in the following figure. The most effective repair method should be adding a backing bracket.

In the left example of the following figure, a crack happens at the heel of the flat bar stiffener. Repair 3 is not recommended at all, because other repairs are either cheaper or stronger. Repair 4 and 5 could be better repair choices. Repair 6 adding a backing bracket and inserting a new plate is robust but may not be very cost effective. To ensure a more robust repair like Repair 6, adding bracket on the both side may be easier than Repair 6.

In the middle example of the following figure, adding bracket on the both side can be considered as an additional repair alternative. If the flat bar stiffener requires to be cropped and part renewed, this may be replaced by a bracket incorporating a soft nose at the longitudinal together with the recommended backing bracket.

In the right example, the crack grows into the longitudinal. Longitudinals not only contribute the hull longitudinal strength but also attach to the shell. While cracks grow into shell, cargo leaking may happen. In order to ensure the water tight integrity of the side shell, it is recommended the fractured longitudinal should be cropped and part renewed as in repair 3 and 6.

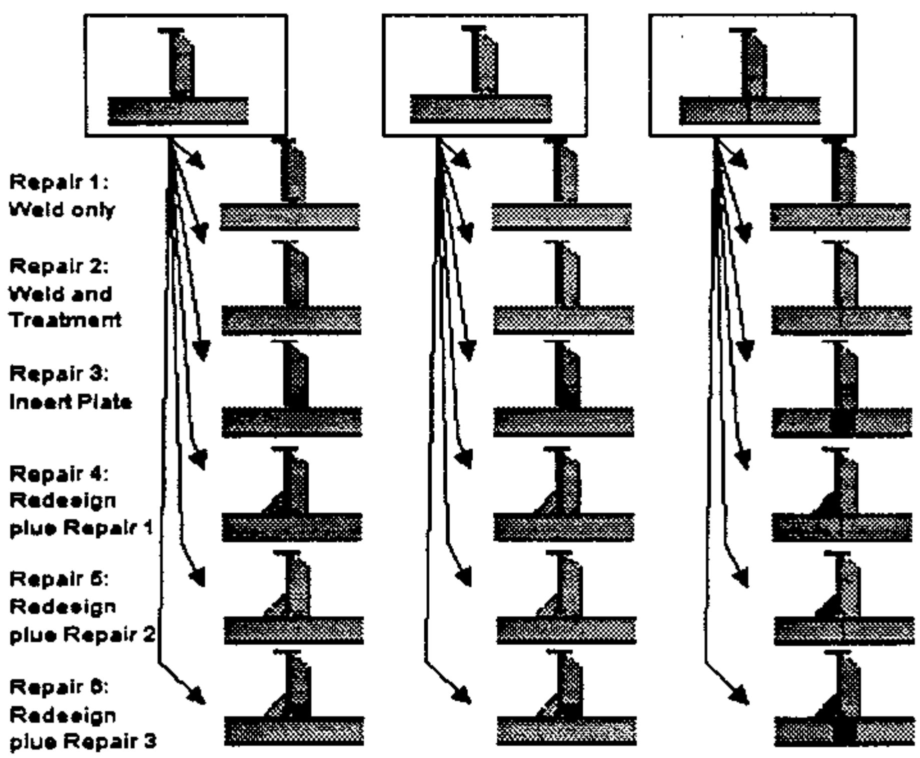


Figure 4.4: Repair alternatives for cracks in flat bar stiffeners

4.2.3 Repairs of Clearance Longitudinal Cutout

Tripping bracket is one of the most critical structural detail, too. There are a variety of configurations of this type. The most common one looks like the those in the following figure. Three typical crack locations are shown here. The left one happens more frequently than the other two. Six repair alternatives are presented in each case. Beside these six, adding a flat bar stiffener or a backing bracket on the longitudinal is another repair alternative.

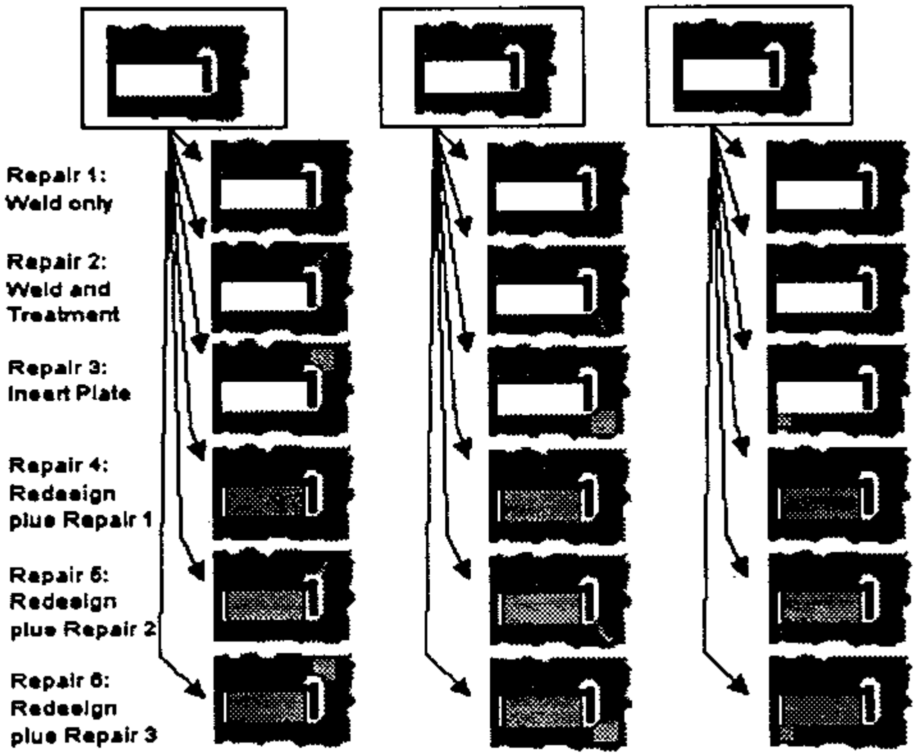


Figure 4.5: Repair alternatives for cracks in longitudinal cutouts

4.3 Repair Decision

Up to now, the most critical aspect of the Repair Management System (RMS) repair evaluation has not been discussed--cost. To be effective, a decision analysis that deals with the uncertainties of the problem and the cost criteria of the owner and operator of the ship is required to help evaluate the optimum repair option. In terms of cost, the optimum repair option is defined as the one that results in the minimum total costs (initial plus future) over the life of the ship (Figure 4.6).

Take the repair of a crack in a longitudinal clearance cutout for an example. There are several repair alternatives available for this failure as described in the previous sections. If the ship is going to be operated for , say, 10 more years, and the owner chooses the cheapest and easiest repair like veeing and welding only. The initial repair cost will be very low. Since this repair is not robust, it may has an expected repair life like, say, less than 2 years only. Then the owner may have to repair it every two years (totally 5 more repairs) if continuing the same repair method. The cost of these 5 more repairs is the future cost and it may be quite high. This is the case of the very right side of Figure 4.6.

In the other hand, if the owner choose a very robust repair like inserting plate and adding a lug, the initial cost may be high. But the expected repair life may be more than 10 years, that means no future repair will be needed. The future repair cost will be as low as zero in this case. This is the case of the very left side of Figure 4.6.

The concept of optimum repair based on the lowest total repair cost is quite simple. There are many factors have to be taken into consideration in the real world. These include available budget, available repair time, out of service cost, and the owner's future plans for the ship. It is difficult to accurately estimate a repair life and cost for a particular repair alternative. There is a large uncertainty associated with the estimation of repair life. It is a challenge to define the optimum or best repair alternative.

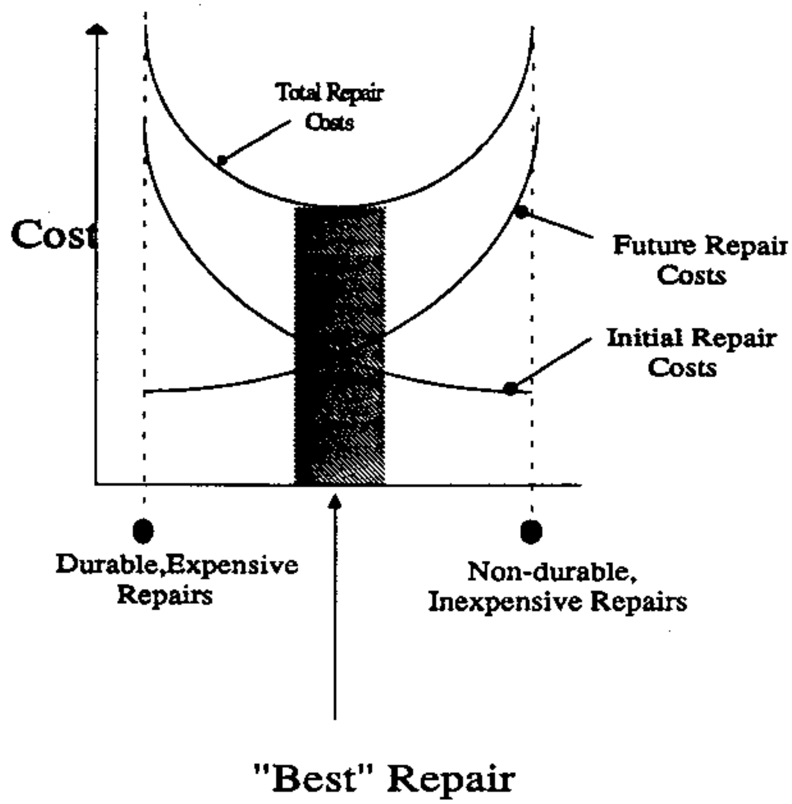


Figure 4.6: Repair Cost Tradeoff

58

CHAPTER 5. REPAIR LIFE ESTIMATION

The key to any repair analysis is the ability to rank repair alternatives according to some index. For the Repair Management System (RMS) the expected life of a repair is used as the index. This index is most useful since time is a critical component in the decision process.

The method of repair life estimations will vary with the mode and cause of failure. For each mode, a different analytical procedure is required. Because ships are plagued primarily by fatigue problems, only the fatigue failure mode is explored in this study.

For quick comparison of repair alternatives as required by the RMS philosophy, it is necessary to adopt an approach that does not rely on lengthy, cumbersome finite-element analysis. The proposed method to be used for the RMS is an approximate method. The method incorporates existing knowledge of material SN curve characteristics (cyclic stress range versus number of cycles to failure curves) and stress concentration factors for critical structural details. An equation which is developed by Wirsching will be used to compute the expected life of a repair on a cracked detail.

5.1 Cumulative Fatigue Damage Model

To evaluate the damage to a detail due the Weibull loading, Miner's rule of cumulative damage is assumed. The accumulation of damage D due to the full range of alternating stresses is approximated by [Wirsching, 1984, 1987]:

$$D = \sum_{i=1}^{n} \frac{N(S_i)}{N_f(S_i)} = \frac{T_f B^m \Omega}{A} \qquad (5.1)$$

$N(S_i)$	=	Number of cycles alternating stress S_i applied
$N_f(S_i)$	=	Number of cycles to failure at stress S_i
T_f	=	Time to failure
B	=	Uncertainty factor in estimation of fatigue stress
Ω	=	Stress parameter, mean
A	=	Life intercept, mean

When the damage is greater than or equal to one, failure is usually assumed to occur. Laboratory tests have shown wide variation in the actual cumulative damage at failure. Defining the damage at failure as Δ_f, the above equation can be rewritten as:

$$T_f = \frac{\Delta_f A}{B^m \Omega} \tag{5.2}$$

For the Weibull stress range model and a single slope SN curve, the stress parameter Ω is given by:

$$\Omega = f_o S_o^m [\ln N_o]^{-(m/\varepsilon)} \Gamma\left(\frac{m}{\varepsilon} + 1\right) \tag{5.3}$$

The average frequency f_0 of the stress cycles is a constant 2.5×10^6 cycles/yr for the wave loading on ship structure. S_0 is the alternating stress that is exceeded on an average of once N_0 cycles. In addition, the mean SN data should be used to remove the bias in the design curves when making comparisons.

To examine how this model can be used to evaluate repairs, consider a crack discovered in 10 years that developed due to high cycle fatigue. Assuming a Weibull parameter and curve designation, the stress range required to produce the failure may be determined. Due to the many assumptions involved, this stress range is only useful when used on a comparative basis. For example, if a crack originating at a cutout corner (C class, m=3.5, log A=14.03, single slope approximation) in the side shell (Weibull parameter 0.9) is discovered in 10 years (T_f=10 years, $f_0 = 2.5 \times 10^6$ cycles/year, $No = f_0 T_f = 2.5 \times 10^7$ cycles), then the calculated peak Weibull stress range to cause failure (D_f=1) based on the mean SN data and no uncertainty (B=1) is:

$$S_o = \frac{(\ln (f_o T_f))^{1/\varepsilon}}{B} \left\{ \frac{\Delta_f A}{f_o T_f \Gamma\left(\frac{m}{\varepsilon} + 1\right)} \right\}^{1/m} = 777 \text{ N/mm}^2 \tag{5.4}$$

If this crack is then ground out and welded up, the SN curve degrades to, say, E class (m=3.0, A=3.29e12), the stress range and Weibull parameter remain the same, and the new mean life to failure T_f (D_f=1) may be estimated by solving the following by iteration for T_f:

$$T_f = \frac{\Delta_f \, A \, [\ln(f_o T_f)]^{(m/\varepsilon)}}{f_o \, (B \, S_o)^m \, \Gamma\left(\frac{m}{\varepsilon} + 1\right)} \Rightarrow T_f = 3.06 \text{ yrs} \tag{5.5}$$

Now the expected mean repair life for the repair of veeing and welding is found to be 3.06 years. Since the veeing and welding may bring potential defects on the weld, the repaired detail inevitably has a shorter life than the original's. This is a fairly reasonable result.

In short, using the Wirsching equation to compute the expected mean repair life of any repair on a particular ship structural detail, four sets of variables are needed (The example values are in the parenthesis):

- SN data of the detail (C curve, m=3.5, A=3.99e12)
- Fatigue life of the detail (T_f = 10 years)
- SN data of the repaired detail (Degraded to E curve, m=3.0, A=3.29e12)
- Stress reduction factor (Stress level is not reduced by this repair.)

The Weibull parameter is unknown and can be assumed to be 0.9. The average frequency f_o of the stress cycles is known to be a constant 2.5×10^6 cycles/yr for the wave loading on ship structure. It can be verified by the following calculation assuming 70 percent ship operation and an average wave encounter period of 9 seconds:

$$\begin{aligned} f_o &= 0.70\left(\frac{1 \text{ cycle}}{9 \text{ sec}}\right)\left(\frac{365 \text{ days}}{1 \text{ year}}\right)\left(\frac{24 \text{ hrs}}{1 \text{ day}}\right)\left(\frac{60 \text{ min}}{1 \text{ hr}}\right)\left(\frac{60 \text{ sec}}{1 \text{ min}}\right) \\ &= 2.5 \times 10^6 \text{ cycles / yr} \end{aligned} \tag{5.6}$$

Whenever the above four sets of information are obtained, the expected repair life can be computed. The following sections will discuss more about how to get these data.

5.2 SN Curve Considerations for Fatigue Failure

To compute the expected repair life, four sets of information are needed as described in the previous section. This section explains how to obtain one of these four, SN data of the detail.

It is very difficult to obtain the capacity (SN class) of a ship structural detail by testing a full scale detail in ship. Therefore laboratory specimens are tested with alternating loading. The relation between the stress range and number of cycle to failure is plotted as a curve. This curve is called SN curve and is assigned a letter (B, C, D, E, F, F2, G, W or others). Different curves represent specimens of different configuration. A ship structural detail can be matched to the S-N curve of a laboratory specimen if they has a similar geometry and loading condition. Different locations within a detail are assigned a SN class that represents the fatigue characteristics of that location.

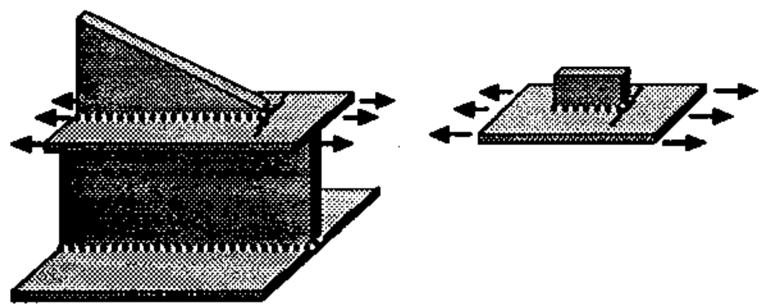


Figure 5.1: A ship structural detail and the corresponding class F fatigue specimen

An indication of the relationship between a ship structure detail and a laboratory fatigue specimen is given in Figure 5.1. The shown fatigue specimen (right side in Figure 5.1) is classified into the class F by the U.K. Department of Energy. Since the ship structural detail shown in the left side of the figure has similar geometry and loading condition as the specimen, the detail can be assigned a SN curve of Class F.

There is an amount of judgment involved in the selection of the appropriate S-N curve for any given case. Work on matching SN curves to ship structural detail has been explored in the past study by the American Bureau of Shipping [Chen, 1992]. The following figure shows some examples in that study.

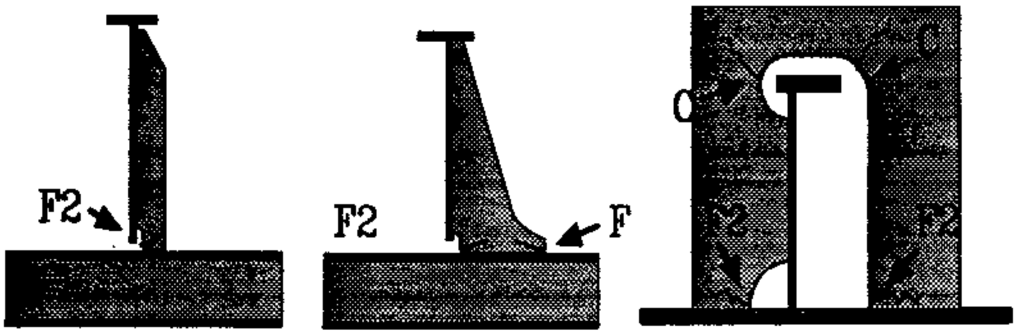


Figure 5.2: S-N class designation on critical structural details

A few organizations, like United Kingdom Department of Energy (DEn) and Department of Civil Engineering of the University of Illinois, have developed sets of SN data. In our Repair Management System, the data from the United Kingdom Department of Energy are used.

Table 5.1 summarizes the design SN curves associated with these designations. SN class designations closer to "A" in the alphabet (i.e., B) represent more durable locations. These curves, which represent the mean data minus two standard deviations (for design purposes) of log N, may be described by:

$$\log N_f = \log A - 2 \log \sigma_{sd} - m \log S = \log A' - m \log S \quad (5.7)$$

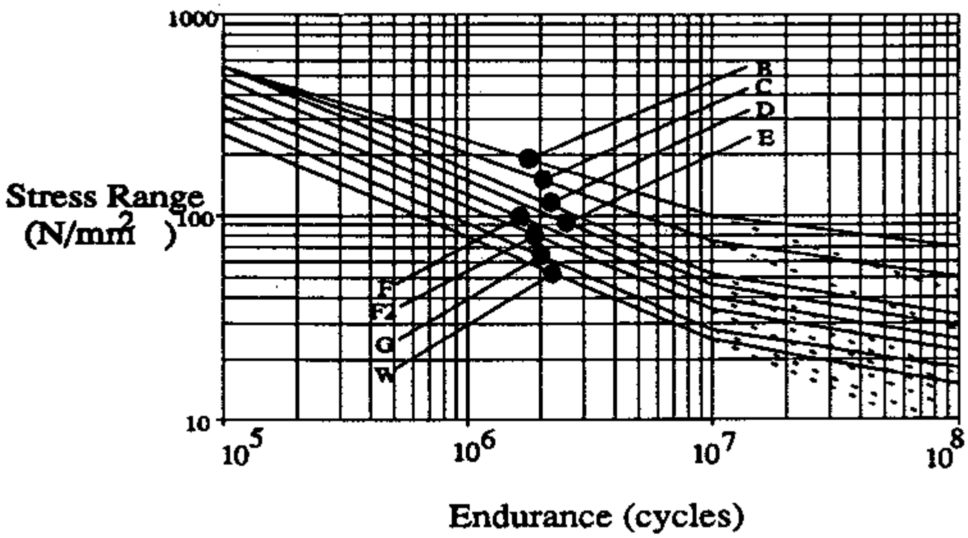
N_f = Predicted number of cycles to failure under stress range S

A = Life intercept

$\log \sigma_{sd}$ = Standard deviation of log N

m = Inverse slope of SN curve

63



	Parameters $N \leq 10^7$			
Curve Class	A (MPa)	A/A'	m	COV of A*
B	2.34 E15	2.29	4.0	0.44
C	1.08 E14	2.54	3.5	0.50
D	3.99 E12	2.63	3.0	0.51
E	3.29 E12	3.14	3.0	0.63
F	1.73 E12	2.74	3.0	0.54
F2	1.23 E12	2.88	3.0	0.56
G	5.66 E11	2.30	3.0	0.43
W	3.68 E11	2.32	3.0	0.44

Table 5.1. Mean SN Curve Constants in Air or Adequately Protected in Seawater

(SN curve plotted above)

[DNV,1984] ,[Wirsching,1987]*

The U.K. DEn specifications provide tables relating to selection of S-N curves for any given structural detail situation.

It should be noted that the SN data scatter very much. Some people use the curve which is two standard deviations below the mean lines. This means that two standard deviations are deducted from mean S-N curve to be on the safe side of test results (See

Figure 5.3), that is, 97.5% survival S-N curve is obtained. In RMS, the mean SN curves are used.

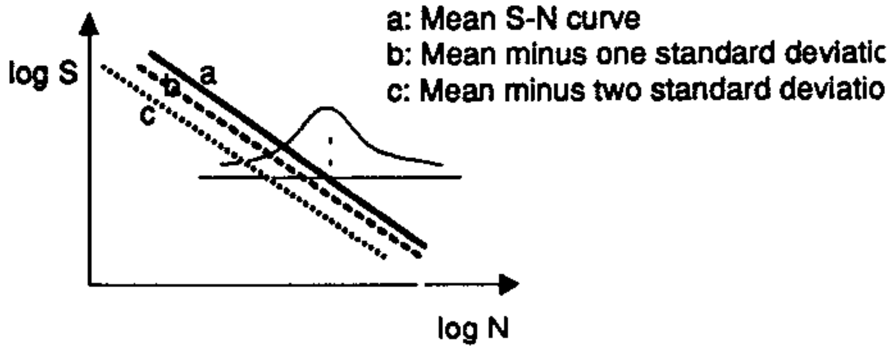


Figure 5.3: S-N curves with different reliability

There is a size effect associated with these curves. To account for this, Equation 5.7 may be modified to the following for all types of welded structure except for butt welds dressed flush and low local bending across the plate thickness:

$$\log N = \log A' - \frac{m}{4}\log\left(\frac{t}{22}\right) - m\log S \qquad (5.8)$$

The variable t is the thickness in millimeters through which a crack will grow (e.g., plate thickness).

S-N performance is also affected by the environment. When steel is subjected to cyclic stresses while in contact with a corrosive environment like sea water, the fatigue strength may be reduced as compared with the fatigue strength for the same number of cycles in air. In tankers, the rules of some class societies now require coating in ballast tanks, so only cargo tanks without coating will potentially suffer this corrosion fatigue.

There are two distinct regions in the figure above Table 5.1. For cycles $N>10^7$ there is a change in slope to model the effect of corrosion. There is some controversy over the actual effect of sea water and cathodic protection on these curves; however, the RMS will allow the SN curve data to be modified to the form desired by the user. For unprotected steel in sea water, a fatigue strength is assumed to be reduced by a factor of 2.0.

5.3 SN Curve of the Repaired Details

As discussed in section 5.1, four sets of information are needed to compute the expected repair life. The first one, SN data of the structural detail, can be obtained as described in section 5.2. The second one, fatigue life of the detail, is the time interval from the delivery of the ship to the time the detail fails. The first two sets of information are fairly easy to obtain. However the third one, SN data of the repaired detail, is not so easy to get.

Consider a crack originating at a cutout corner in the transverse web, a C class of SN curve will be assigned. If the crack is repaired by veeing and welding, the capacity of that location will be lower due to the potential defects in the weld. The SN class after repair could be some curve lower than C curve. Until now no experiment has been carried out to designate a SN class for this kind of repair. Similarly if the crack is repaired by veeing and welding plus post weld improvement, the appropriate SN curve for the repaired location is also unknown. Only while the repair is done by inserting a new plate the SN curve is sure to be the same curve as the original one, since the geometry and the material of the detail stay the same.

To fix the unavailability of the SN information, the RMS temporarily assume the SN curve will be lowered by two classes after repairing by veeing and welding plus post weld improvement. It will be lowered by one class after repairing by veeing and welding only. Also the data in the file is designed to be easy to change by users. Users can update them while new information is available.

This problem is discussed more in details in Chapter 8 of this report. A new research project has been proposed to solve this problem. This project will be continued during the period 1993 - 1994.

5.4 Stress Concentration Factor Considerations

Fatigue is dependent on the local stress in a critical structural details. The local crack opening stress may be estimated either by detailed finite element analysis or through the intelligent use of stress concentration factors. Stress concentration factors (SCF) have been developed for various structural details based on both testing and finite-element analysis results. A stress concentration factor is defined mathematically by:

$$K = \frac{\sigma}{\sigma_n} \tag{5.9}$$

σ = Concentrated stress level

σ_n = Nominal stress level

For a ship structural detail, the nominal loadings may be broken up into longitudinal stress due to hull bending (vertical and athwart ship), shear (vertical), and net external pressure. For a complete description of the stress concentration factors from a finite element analysis model, each of these load cases should be applied independently to the part. The results from each of these analyses can then be used to complete a table of stress concentrations that is a function of the detail configuration, the location within the detail, and the applied stress direction. An example of these factors is shown in Table 5.2.

These stress concentrations should be expressed in terms of the tensile stress normal to the expected direction of cracking since typically we deal with Mode I cracking (resulting from tensile stress). A negative stress concentration could be used to represent a reversal between applied nominal stress and the stress at the crack location. Careful consideration of the restraints on the model is also required for all loading cases. When new details are analyzed by finite element methods or by testing, results can be stored in this tabular format for immediate use in the evaluation of repairs.

Depending on the location of the detail within the ship, the effect of these stress concentrations will vary. For example, around the waterline location of the ship, the stress

$$S_o' = S_o\left(\frac{K_{repair}}{K_{original}}\right)\left(\frac{t_{original}}{t_{repair}}\right)^n \quad (5.11)$$

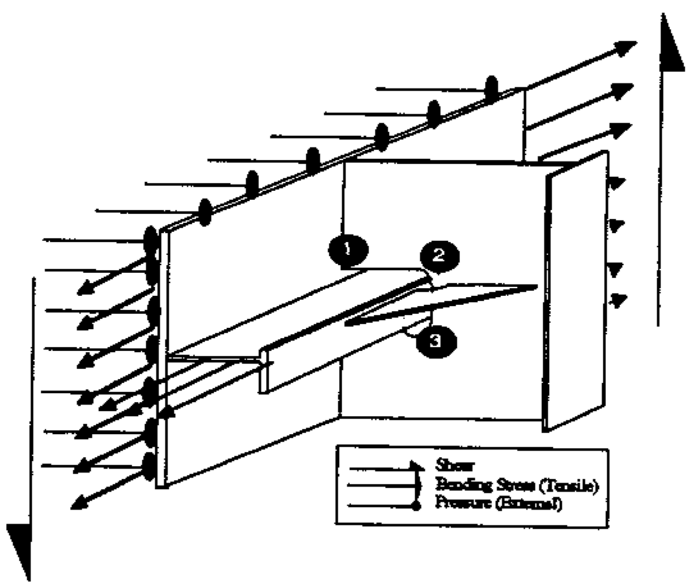
K = Stress concentration factor of the repaired and original detail

t = Thickness of the repaired and original detail

n = Factor which is dependent on the dominant stress direction

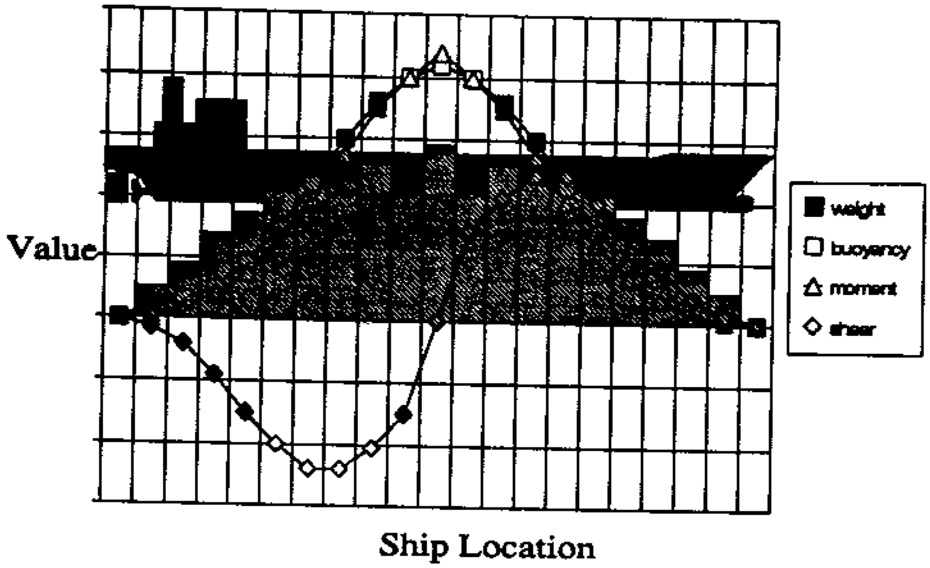
The term, $K_{repair}/K_{original}$ is a stress reduction factor. In stead of storing the complex table of SCFs, RMS 2.0 stores the stress reduction factors only to simplify the data file.

As the actual performances of repairs are evaluated and additional analyses are completed, the stress reduction factors (or alternatively SCFs and load ratios) could be continually updated, resulting in more accurate repair life estimations.



Location	Load Case			
	1	2	3	4
	Vertical Bending	Athwart Bending	Pressure	Shear
1	K_{11}	K_{12}	K_{13}	K_{14}
2	K_{21}	K_{22}	K23	K_{24}
3	K_{31}	K_{32}	K_{33}	K_{34}

Table 5.2. Stress Concentration Factors K, Side Shell Detail A [Keith, 1992]



		Load Case			
		1	2	3	4
Fore/Aft Location	**Vertical Location**	Vertical Bending	Athwartship Bending	Pressure	Shear
Forward 1/3	Top 1/3	.5	.5	1	0
	Mid 1/3	0	.5	1	1
	Lower 1/3	.5	.5	1	0
Amidships	Top 1/3	1	1	0	0
	Mid 1/3	0	1	1	.5
	Lower 1/3	1	1	.7	0
Aft 1/3	Top 1/3	.5	.5	0	1
	Mid 1/3	0	.5	1	0
	Lower 1/3	.5	.5	.7	1

Table 5.3. RMS Expert Load Ratios for Side Shell Structure Due to Ship Location [Keith, 1992]

(typical hogging load distribution shown above)

5.5 Weibull Loading Model

To evaluate a component for fatigue, the alternating stress level must be determined. The effect of mean stress can generally be ignored due to its small influence on the fatigue strength of steels [ISSC, 1988, 1991]. Several models can be used to represent the long term stress range, including wave exceedance diagrams, spectral methods, the Weibull model and the Nolte-Hansford model. A Weibull model to represent the long term distribution of cyclic stress ranges will be used for the RMS due to its relative simplicity. Using the Weibull model, the alternating stress in ship structure is represented by:

$$F(S) = Pr(s > S) = \exp\left(-\left(\frac{S}{\delta}\right)^{\varepsilon}\right) \tag{5.12}$$

F(S)	=	Probability that stress range S is exceeded
ε	=	Weibull shape parameter
δ	=	Weibull scale parameter

The scale parameter d may be related to the stress range and the return period N_0 by:

$$\delta = \frac{S_o}{(\ln N_o)^{1/\varepsilon}} \tag{5.13}$$

S_0 is the alternating stress that is exceeded on an average of once every N_0 cycles (design life or actual life in cycles). So now we have a one parameter distribution represented by:

$$F(S) = Pr(s > S) = \exp\left(-\left(\frac{S}{S_o}\right)^{\varepsilon} \ln N_o\right) \tag{5.14}$$

Defining N as the number of stress variations of N_0 that exceed S this equation may be expressed as:

$$S = S_o\left(1 - \frac{\log N}{\log N_o}\right)^{\frac{1}{\varepsilon}} \tag{5.15}$$

The Weibull shape parameter ε will vary with the environment (trading route, sea conditions) and the response of the ship structure to the environment. Specifically, ε will vary with ship length, ship type, location within the ship and the trading route under operation. For crude carriers and cargo ships ε is typically between 0.7 and 1.3 [Munse, 1981]. However it is currently assumed to be 0.9 in RMS 2.0 no matter where the crack is. The Weibull parameter may be obtained more accurately by direct instrumentation or detailed wave and structural analysis.

5.6 Procedures of Computing Repair Life

When a repair is made, a combination of three things can occur:

1. a change in the SN curve designation of a location due to modifications such as welding;
2. a change in the stress concentration factor (thus alternating stress level) of a location due to change in geometry; and/or
3. a change in component thickness (thus alternating stress level) due to the addition of a thicker insert plate or doubler.

To compare repair alternatives, these three changes must be accounted for.

First, N_O is assumed to be life at inspection. For example, if a crack is discovered at a ship life of 10 years then:

$$N_o = 10\ f_o = 10 \text{ years} \left(\frac{2.5 \times 10^6 \text{ cycles}}{1 \text{ year}} \right) = .25 \times 10^8 \text{ cycles} \qquad (5.16)$$

Second, a best estimate of S_O to cause failure based on the SN curve designation, the Weibull shape parameter and the cumulative damage approach is calculated by the following:

$$S_o = \frac{(\ln N_o)^{1/\varepsilon}}{B} \left\{ \frac{\Delta_f A}{f_o T_f \Gamma\left(\frac{m}{\varepsilon} + 1\right)} \right\}^{1/m} \qquad (5.17)$$

due to vertical bending is minimal (close to the neutral axis) and the stress due to external pressure is very high (wave loading). Therefore, to compare the stress levels at various locations within several repair alternatives, we must develop a table of the relative magnitudes of the loadings as a function of the location within the ship.

While the geometry is modified, we have a change in stress level at the crack location. The change in stress level is determined by the load ratio in Table 5.2 and the stress concentration factors for the original and modified details at the crack location, Table 5.3. The overall stress concentration factor for both the original and modified detail is determined as:

$$K_{combined} = \sum_{i=1}^{n} K_{ij} R_j \tag{5.10}$$

i = Location number on the detail

j = Load case number

n = Total number of load cases

K_{ij} = Stress concentration factor for load case i at detail location j

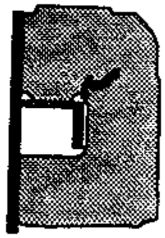
R_j = Load ratio for load case j at the ship location under study.

A linear combination is valid only if stress concentration factors are defined normal to the crack direction and not in terms of combined stresses.

Table 5.2 summarizes these expert load ratios for the RMS based on "typical" moment and shear diagrams as illustrated above the table. Since the process of identifying the local loads through wave spectrum and global structural analysis is too tedious, the data in the table is calibrated based on expert opinions. The maximum value of one for a given load case represents the ship location of maximum load contribution. A more detailed loading library for future use might account for a finer definition of the location in the ship, the size of the ship, trading route, the beam approximation of the ship and other factors to get a more accurate estimate of the loading variation.

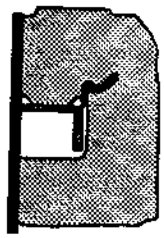
The two SCFs (original and repaired) will be needed by the following equation to correct for changes in stress concentration factors in the repaired detail:

modification of the SN curve from C to E class. Following the computing procedures of the previous section, the result of the expected repair life is about 3.06 years only.



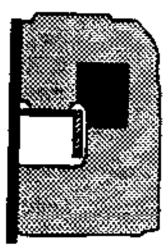
Repair 2 Vee and Weld Plus Post Weld Improvement

This repair is almost the same as Repair 1. Since the weld surface is improved, the material degradation due to welding is accounted for by the modification of the SN curve from C to D class. Following the computing procedures of the previous section, the result of the expected repair life is about 3.89 years only.



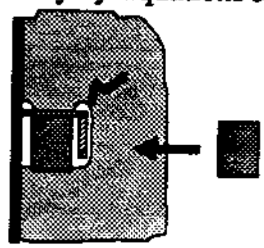
Repair 3 Insert a New Plate:

The geometry of this detail has not been modified, but the insert plate thickness may be different from the original plate. At the crack location, the expected life of this repair is assumed to be equal to that of the original, that is, 10 years. In case that the plate thickness t is modified, the new stress range should be estimated by Equation 5.18. A better repair can be obtained. Notice that two new hot spots are introduced by the weld around the inserted plated. At the weld locations, a combination of a stress concentration factor increase due to the change in plate thickness and a change in the SN curve due to the addition of the weld occurs. Therefore the inserted plate should be carefully configured to avoid new hot spots.



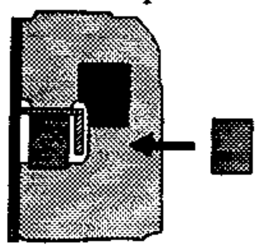
Repair 4 Vee & Weld Plus Add a Lug:

In this case the geometry has been modified so that we have a change in stress level plus a change in SN curve designation at the crack location. The change in stress level is determined by a stress reduction factor. Notice that the SN curve has been degraded at the lug weld location and at the location of the crack, too. Each of these locations should be evaluated separately by Equations 5.18 and 5.19.



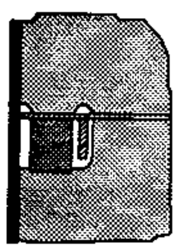
Repair 5 Insert Plate Plus Add a Lug:

In this case the geometry has been modified so that we have a change in stress level plus a change in SN curve designation at the weld locations. There is no change in the SN curve at the original crack location, but possibly a change in plate thickness of the inserted plate. Evaluation continues as for Repair 4.



Repair 6 Add a Bracket Plus Repair 5:

In this case the geometry has been modified beyond repair 4 with the addition of brackets. Evaluation continues as for Repair 5.



A simplified approach to the estimation of the fatigue life of repair alternatives has been outlined and demonstrated for a typical side shell structural detail. Depending on the data available, some required information might be missing to estimate the repair life. The RMS should report this missing data and allow for easy addition of any new results to the knowledge-base and database.

CHAPTER 6. RMS DATABASE

Through a ship's life, a number of surveys will be carried out. Thousands of pieces of data on coatings, fractures, and gaugings will be recorded in each survey. Due to the amount of survey data, the data are usually difficult and expensive to record, retrieve and analyze. The data can consist of rough sketches in a repair superintendent's notebook and shipyard invoices collected in a repair file. Data that resides in the experience of individuals involved in ship maintenance also needs to be archived. The gathering, storage, retrieval, and analysis of the huge quantity of the information can be facilitated by developing a computer database system. Database systems can significantly improve the efficiency and effectiveness of ship maintenance. Development of maintenance plans, specifications, and reports can be greatly facilitated with the help of such systems. In general, database systems are not well developed in the ship industry compared with those of other industries. Some industrial organizations have pioneered the development of computer based database systems. At the present time, these systems are still in their early stages of development. The RMS 2.0 includes a simple but powerful graphical crack database.

6.1 RMS Database System

The general objectives of an RMS database system development are as follows:

- Collect meaningful data.
- Store the data.
- Provide means for logical data management.
- Provide access to the relevant data easily.
- Allow for the organization of the data in a form suitable for analyses.
- Analyze the data.
- Show trends of the data.
- Communicate and report the data.

Some of these objectives have been fulfilled in the RMS 2.0. It is possible to reach all of them. However it will take more time and effort to complete it.

The overall advantage of such a comprehensive database system is that the data are in electronic format so that the data can be transferred easier and faster by modems or floppy diskettes. The data can be transmitted among ship owners, shipyards, repair yards, design offices via telephone and satellite communication. It also can enhance the efficiency of inspection, maintenance, and repair by eliminating manual writing of the steel repair specification or manual drafting of repair drawings. In addition, it provides the capacity to quickly update corrosion, fatigue, and repair databases.

6.2 Current Database Developments

Three database system and one plan that have been developed before RMS 2.0 are reviewed here.

6.2.1 Corrosion Databases

A corrosion database was created in Ship Structural Maintenance Project (SMP) at U. C. Berkeley [Pollard, 1991]. A total number of about 7,200 gauging data has been input into the database manually. The purpose of this database is to calculate the corrosion rates of different tank types, detail types or locations. The database can compute the means and the standard deviations of corrosion rates. The corrosion rates of four tank types, twenty two detail types, and nine locations were calculated. A database management system was developed in the corrosion database to facilitate easy data entry and provide flexible data analysis. The database management system provides a user friendly screen to facilitate data input, analyses, and evaluations of the information.

It is not easy to create a corrosion database. A particularly difficult part of the development of the corrosion databases is the problem associated with the very large volumes of data that must be recorded and input to the computer. Generally, a single gauging survey can result in 8,000 to 10,000 readings. These readings have to be recorded on paper. However, paper based recording procedures are very labor intensive. Upon completion of the survey, the inspector has to transcribe the information to a smooth form for others to take appropriate action. It can result in long lag-times between when the data is gathered and evaluated. This result in substantial inefficiencies during the maintenance and repair operations.

Another Problem is that there is no standard way to describe the location of a particular survey result. There is no standard coordinate system. The precise spatial location of inspection results within a hull structure is difficult during the conduct of the inspections. Development of graphical data reporting and recording formats will help gathering, verifying and reporting such information.

6.2.2 Fatigue Cracking Databases

A fatigue crack database has been created in Ship Structural Maintenance Project at U. C. Berkeley [Schulte-Strathaus and Bea, 1991]. The fatigue crack data of 10 VLCCs were provided by the SMP participants. A total number of 3584 cracks has been input into the database.

This database serves the following purposes:

- Provide a mean for the intelligent management of fatigue crack data.
- Provide insight about where to look for cracks and thus also enhance the effectiveness of ship inspection.
- Provide the mean for statistic analysis of crack locations and show trends.
- Show relative percentage of fatigue cracks for a certain type of details, and thus identify what types of details crack most frequently.

Again, there is no standard way to describe the location of a particular survey result. There is no standard coordinate system. The precise spatial location of inspection results within a hull structure is difficult during the conduct of the inspections. Development of graphical data reporting forms may help gathering such information.

In this database the location of a crack is determined as follows. The longitudinal position is obtained by including the frame number. For the vertical position on the side shell, the longitudinal bulkhead and the transverse bulkheads the ship has been divided into three equally spaced zones, low , middle, and top thirds. This procedure allows one to compare different ships. The division into three zones was considered to be practical and sufficient for the desired degree of accuracy. The same zones have been used in the corrosion database. The horizontal position is defined with regard to port and starboard and again by the zones, which show, whether a crack is on the side shell, the longitudinal bulkhead or the transverse bulk. A further division in the horizontal direction was omitted as in the corrosion database where the omission was made for keeping the amount of input to a minimum.

In addition to the locations of cracks, the description and the geometry of the occurring cracks has to be defined. Since one detail, say, side shell longitudinal connection to web frame is very likely to be different from one shipyard to another. This fact makes it very difficult to describe the geometry of a cracked detail without the use of very detailed drawings. In the CATSIR database this problem is solved by relating the included information to CAD drawings, which can be seen on the screen and also be used for data input. This approach is considered to be very promising.

The database of SMP did not adopt the idea of graphical database, because the data input and the setup of a new drawing for a new crack can result in higher cost for the owners and operators of the VLCC's. Instead, a set of keywords has been established, which allows a description of the cracked detail. These keywords also allow statistical analysis of the input data since they have a fixed format and can be used to sort the data. The information available when using this approach is less detailed, but it has the advantage that less data input is required and the keywords are easily memorized. These keyword is shown in Table 6.1 for longitudinal members.

The CAIP will, in the future, require management of the vast amount of

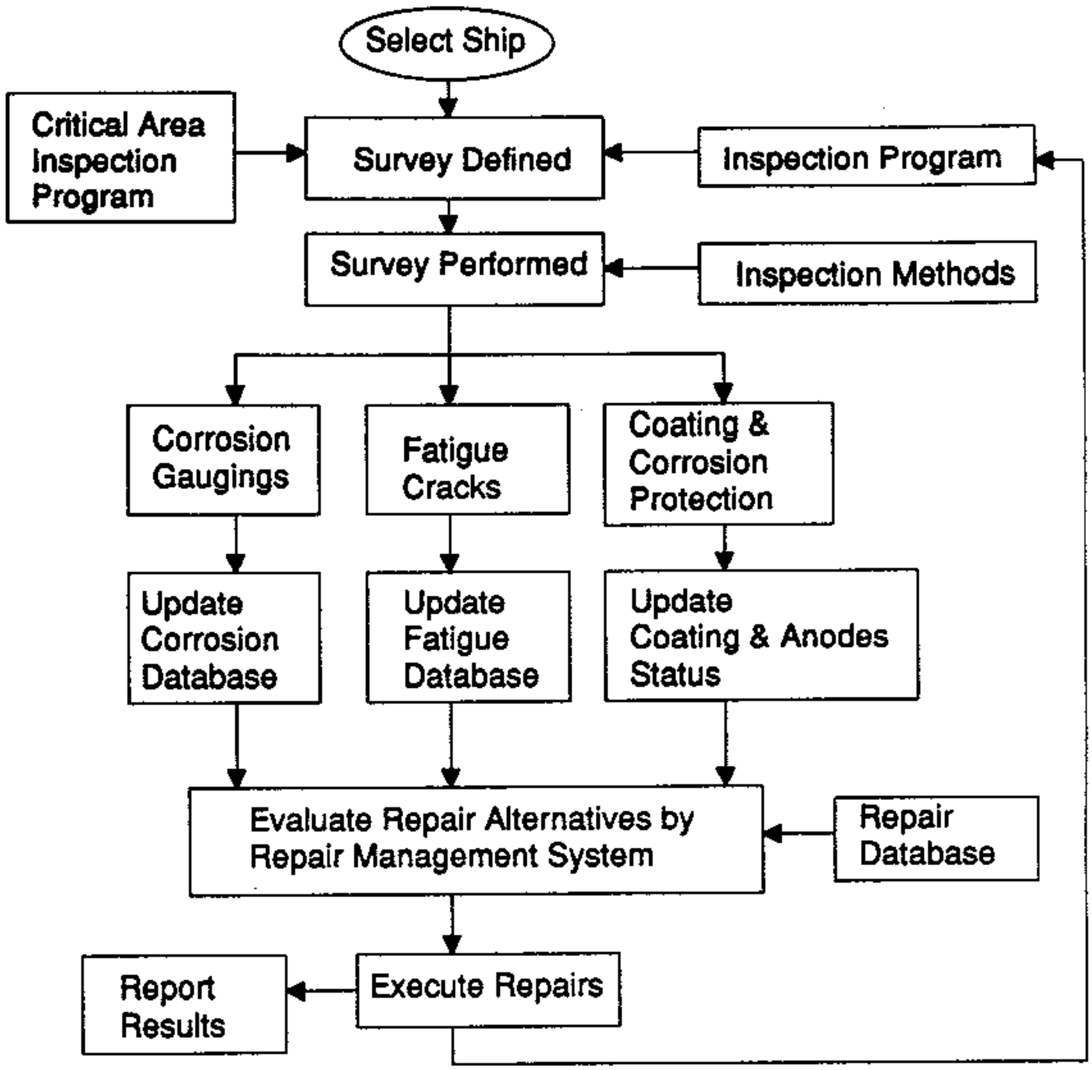


Figure 6.1: Basic parts of RMS system for inspection, maintenance, & repair

Figure 6.1 shows the basic parts of a RMS flow chart for inspection, maintenance and repair. Once a ship is ready for service, a series of surveys can be scheduled according the inspection program. The objective and scope of the internal structural inspections are defined. The access methods and data recording methods are chose, and then the survey is performed. The survey results including corrosion gaugings, fatigue cracks, status of coating and corrosion protection system, or other structural defects are updated into the corresponding databases. Using the survey data, a Repair Management System evaluates repair alternatives. Finally the repairs are carried out.

CHAPTER 6. RMS DATABASE

Through a ship's life, a number of surveys will be carried out. Thousands of pieces of data on coatings, fractures, and gaugings will be recorded in each survey. Due to the amount of survey data, the data are usually difficult and expensive to record, retrieve and analyze. The data can consist of rough sketches in a repair superintendent's notebook and shipyard invoices collected in a repair file. Data that resides in the experience of individuals involved in ship maintenance also needs to be archived. The gathering, storage, retrieval, and analysis of the huge quantity of the information can be facilitated by developing a computer database system. Database systems can significantly improve the efficiency and effectiveness of ship maintenance. Development of maintenance plans, specifications, and reports can be greatly facilitated with the help of such systems. In general, database systems are not well developed in the ship industry compared with those of other industries. Some industrial organizations have pioneered the development of computer based database systems. At the present time, these systems are still in their early stages of development. The RMS 2.0 includes a simple but powerful graphical crack database.

6.1 RMS Database System

The general objectives of an RMS database system development are as follows:

- Collect meaningful data.
- Store the data.
- Provide means for logical data management.
- Provide access to the relevant data easily.
- Allow for the organization of the data in a form suitable for analyses.
- Analyze the data.
- Show trends of the data.
- Communicate and report the data.

Some of these objectives have been fulfilled in the RMS 2.0. It is possible to reach all of them. However it will take more time and effort to complete it.

The overall advantage of such a comprehensive database system is that the data are in electronic format so that the data can be transferred easier and faster by modems or floppy diskettes. The data can be transmitted among ship owners, shipyards, repair yards, design offices via telephone and satellite communication. It also can enhance the efficiency of inspection, maintenance, and repair by eliminating manual writing of the steel repair specification or manual drafting of repair drawings. In addition, it provides the capacity to quickly update corrosion, fatigue, and repair databases.

6.2 Current Database Developments

Three database system and one plan that have been developed before RMS 2.0 are reviewed here.

6.2.1 Corrosion Databases

A corrosion database was created in Ship Structural Maintenance Project (SMP) at U. C. Berkeley [Pollard, 1991]. A total number of about 7,200 gauging data has been input into the database manually. The purpose of this database is to calculate the corrosion rates of different tank types, detail types or locations. The database can compute the means and the standard deviations of corrosion rates. The corrosion rates of four tank types, twenty two detail types, and nine locations were calculated. A database management system was developed in the corrosion database to facilitate easy data entry and provide flexible data analysis. The database management system provides a user friendly screen to facilitate data input, analyses, and evaluations of the information.

It is not easy to create a corrosion database. A particularly difficult part of the development of the corrosion databases is the problem associated with the very large volumes of data that must be recorded and input to the computer. Generally, a single gauging survey can result in 8,000 to 10,000 readings. These readings have to be recorded on paper. However, paper based recording procedures are very labor intensive. Upon completion of the survey, the inspector has to transcribe the information to a smooth form for others to take appropriate action. It can result in long lag-times between when the data is gathered and evaluated. This result in substantial inefficiencies during the maintenance and repair operations.

Another Problem is that there is no standard way to describe the location of a particular survey result. There is no standard coordinate system. The precise spatial location of inspection results within a hull structure is difficult during the conduct of the inspections. Development of graphical data reporting and recording formats will help gathering, verifying and reporting such information.

6.2.2 Fatigue Cracking Databases

A fatigue crack database has been created in Ship Structural Maintenance Project at U. C. Berkeley [Schulte-Strathaus and Bea, 1991]. The fatigue crack data of 10 VLCCs were provided by the SMP participants. A total number of 3584 cracks has been input into the database.

This database serves the following purposes:

- Provide a mean for the intelligent management of fatigue crack data.
- Provide insight about where to look for cracks and thus also enhance the effectiveness of ship inspection.
- Provide the mean for statistic analysis of crack locations and show trends.
- Show relative percentage of fatigue cracks for a certain type of details, and thus identify what types of details crack most frequently.

Again, there is no standard way to describe the location of a particular survey result. There is no standard coordinate system. The precise spatial location of inspection results within a hull structure is difficult during the conduct of the inspections. Development of graphical data reporting forms may help gathering such information.

In this database the location of a crack is determined as follows. The longitudinal position is obtained by including the frame number. For the vertical position on the side shell, the longitudinal bulkhead and the transverse bulkheads the ship has been divided into three equally spaced zones, low , middle, and top thirds. This procedure allows one to compare different ships. The division into three zones was considered to be practical and sufficient for the desired degree of accuracy. The same zones have been used in the corrosion database. The horizontal position is defined with regard to port and starboard and again by the zones, which show, whether a crack is on the side shell, the longitudinal bulkhead or the transverse bulk. A further division in the horizontal direction was omitted as in the corrosion database where the omission was made for keeping the amount of input to a minimum.

In addition to the locations of cracks, the description and the geometry of the occurring cracks has to be defined. Since one detail, say, side shell longitudinal connection to web frame is very likely to be different from one shipyard to another. This fact makes it very difficult to describe the geometry of a cracked detail without the use of very detailed drawings. In the CATSIR database this problem is solved by relating the included information to CAD drawings, which can be seen on the screen and also be used for data input. This approach is considered to be very promising.

The database of SMP did not adopt the idea of graphical database, because the data input and the setup of a new drawing for a new crack can result in higher cost for the owners and operators of the VLCC's. Instead, a set of keywords has been established, which allows a description of the cracked detail. These keywords also allow statistical analysis of the input data since they have a fixed format and can be used to sort the data. The information available when using this approach is less detailed, but it has the advantage that less data input is required and the keywords are easily memorized. These keyword is shown in Table 6.1 for longitudinal members.

Table 6.1: Code for locations of longitudinal members [Schulte-Strathaus and Bea, 1991]

Longitudinal Members		Code
Deck Plating		DP
Bottom Plating		BP
Inner Bottom Plating		IBP
Side Shell Plating		SP
Longitudinal Bhd Plating		LBP
Deck Longitudinals	Web	DLW
	Flange	DLF
	Bracket	DLB
Bottom Longitudinals	Web	BLW
	Flange	BLF
	Bracket	BLB
Inner Bottom Longitudinals	Web	IBLW
	Flange	IBLF
	Bracket	IBLB
Side Longitudinals	Web	SLW
	Flange	SLF
	Bracket	SLB
Longitudinal Bhd Longitudinals	Web	LBLW
	Flange	LBLF
	Bracket	LBLB
Deck (Longl.) Girders	Web	DGW
	Face Plate	DGF
	Bracket	DGB
Bottom (Longl.) Girders	Web	BGW
	Face Plate	BGF
	Bracket	BGB
Side (Longl.) Girders	Web	SGW
	Face Plate	SGF
	Bracket	SGB
Longl. Bhd (Longl.) Girders	Web	LBGW
	Face Plate	LBGF
	Bracket	LBGB
Center (Longl.) Girders	Web	CGW
	Face Plate	CGF
	Bracket	CGB

The CAIP will, in the future, require management of the vast amount of information being accumulated. Thus, a computerized database system can be used for typical defect documentation and inspection results. From the database, trends and critical areas can be determined as required by the CAIP. However, not all ship owners use computers to manage the information obtained during a survey at the present time.

6.2.5 CATSIR System

The procedures for collecting, handling, interpreting and gauging inspection data have remained little changed over the years. An ultrasonic gauging team of two to four men would board the vessel, take gauging in the tanks, record them in a notebook, and then at the end of the day, transpose them to a draft report. It generally takes two to three weeks to complete such a survey. After leaving the ship, the team would return to their office and again transpose the data, combine it with drawings and photographs that had been taken and prepare a final report. An engineer would sort through the data and compare the gauging readings with the original thickness and wastage allowances. The areas of steel to be replaced and the surfaces to be coated are then decided. The periodic overhaul specifications and drawings are prepared manually. The whole process is time consuming and requires a lot of labor.

To improve the efficiency of the inspection and maintenance process, the basis for a comprehensive database system has been developed by Chevron Shipping. The PC-based computer information system is identified as CATSIR (Computer Aided Tanker Structure Inspection and Repair) which combines a data base program and AUTOCAD, a computerized drafting program [Ternus, 1991][Tikka and Donnelly, 1991]. It has been under development since 1986.

To use CATSIR, the gauging team personnel enters inspection information and gauging data into the CATSIR database while they are on the ship. The hull structure drawings, together with the steel grade and original thickness for each element of the structure, can be stored in the AUTOCAD program before the survey. The engineer who interprets the gauging data and decides the required maintenance can display the structural drawing for any part of the ship's tank structure on the computer screen. Annotated comments with the display contain the general inspection information. The gauging data itself is annotated at the appropriate location on the drawing. If it is decided to replace the

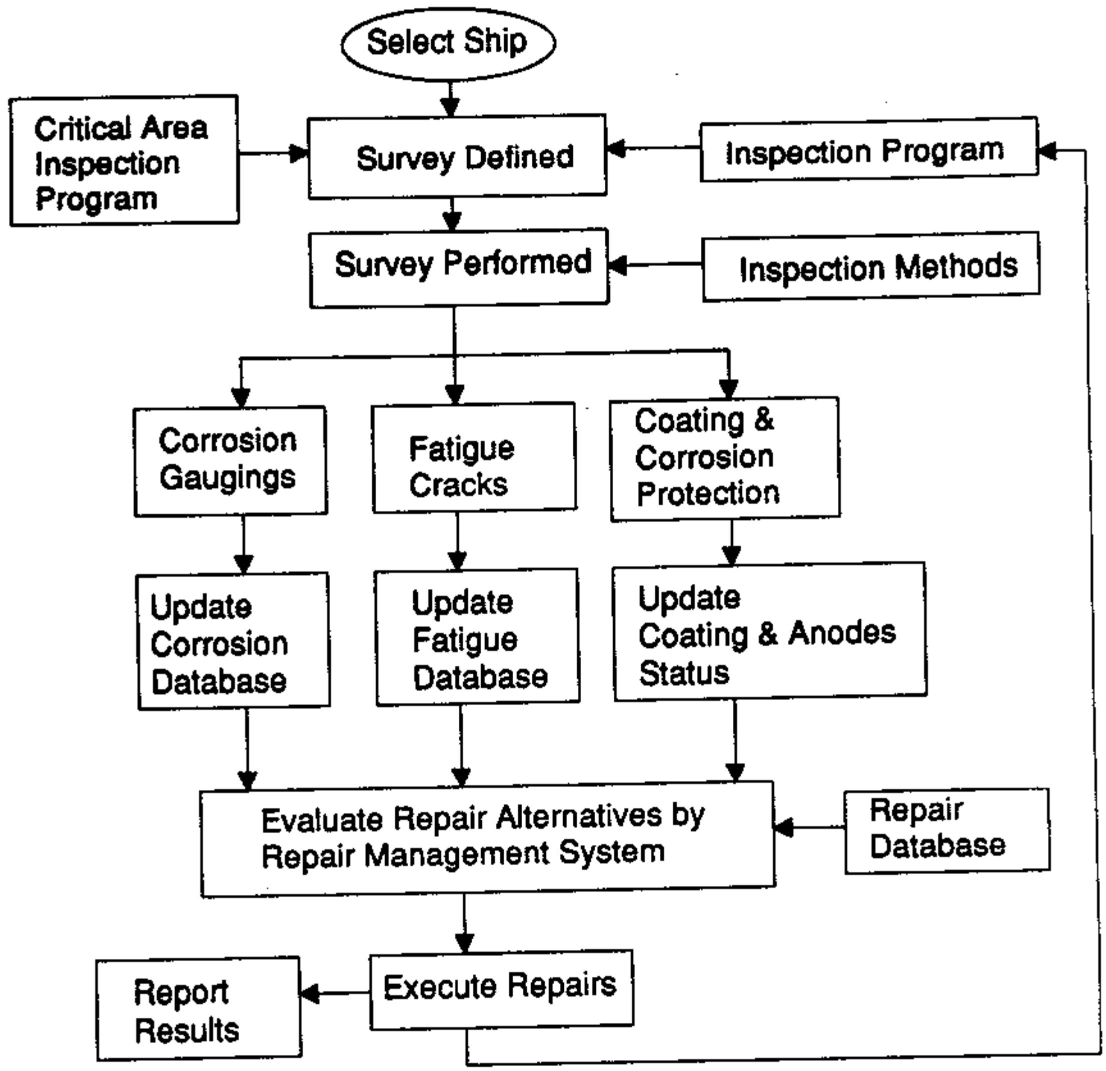


Figure 6.1: Basic parts of RMS system for inspection, maintenance, & repair

Figure 6.1 shows the basic parts of a RMS flow chart for inspection, maintenance and repair. Once a ship is ready for service, a series of surveys can be scheduled according the inspection program. The objective and scope of the internal structural inspections are defined. The access methods and data recording methods are chose, and then the survey is performed. The survey results including corrosion gaugings, fatigue cracks, status of coating and corrosion protection system, or other structural defects are updated into the corresponding databases. Using the survey data, a Repair Management System evaluates repair alternatives. Finally the repairs are carried out.

coating in a certain area, the area can be outlined with a cursor and the program will calculate the number of square meters of coating required. Alternatively if it is decided to renew that part of the structure, the program will calculate the number of pounds of steel required. The database is then updated to include the required repairs.

By using a database like CATSIR, ship owners can develop a cooperative program with some repair yards which are aimed at producing high quality repairs . Each of shipyards has the same database program so that information regarding the steel and coating work is submitted via computer disk. The shipyards can use the program to produce drawings for the repair shops indicating where steel is to be renewed and coating replaced. This allows the yard to plan the work before the ship arrives so as to minimize interference between crafts.

In summary, CATSIR has the following advantages:

1. It improves the productivity of the gauging team by eliminating the draft report and simplifying the final report. The final report consists of a floppy disk containing the gauging information and the comments regarding the vessel inspection.
2. It improves repair planning productivity by eliminating manual writing of the steel repair specification and by automatically calculating steel quantities and coating areas. It also eliminates manual drafting of repair drawings and provides the capability to quickly update repair specifications and drawings in the field.
3. It enhances the efficiency and quality of the inspection and repair. The inspection team and the repair team can both communicate with the home office naval architect, transmitting copies of the information contained on the floppy disks via satellite communications. Naval architects in the home office can then participate in decisions to modify the inspection program or to change the repair specification.
4. CATSIR provides a "one-stop" data bank for all of the tanker structural maintenance data. The analyses of trends are facilitated by sorting data in the data base to collect and display gauging data, which has been obtained over a number of years, from the same location.

6.3 RMS Crack Database

A crack database has been developed in the RMS 2.0. This version of RMS crack database can only handle only one failure mode, fatigue crack. It can store the general information of ship, a ship three view layout, three classes of crack on the ship layout, a crack detail information and the drawing of cracked structural details. Since it is a prototype of graphical database, it can be further developed into a powerful database with ability to handle corrosion gaugings, fatigue crack, coating status, and any information needed by ship maintenance.

6.3.1 Ship Data

This database has three pre-defined ship layouts including a single hull tanker, a bulkcarrier and a container ship (see Figure 6.2). The three layouts has most typical configurations such that most user can simply adopt them as their ship drawings. It also provides a user input mode to allow users import their own ship drawings. By this way all ship types can utilize the functions of RMS as long as they have their ship drawings. The ship drawing can be scanned into a bitmap file easily or users can draw them by using some MS-Windows based drawing programs.

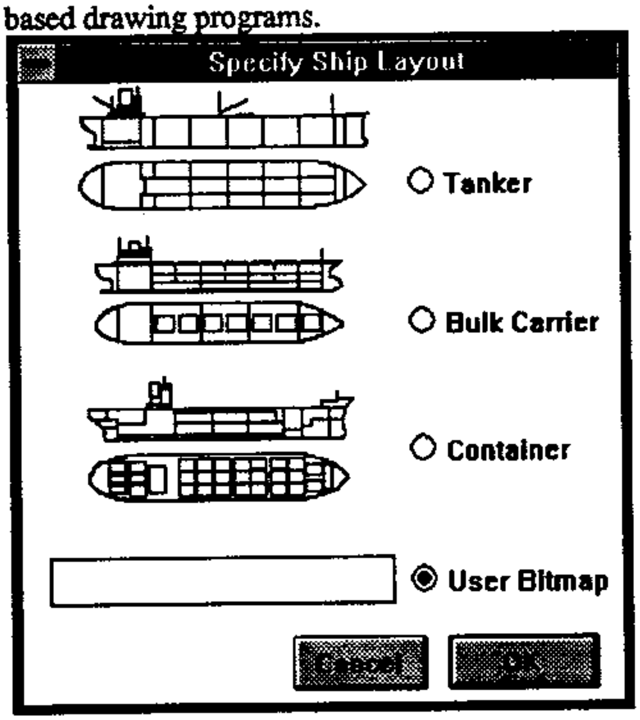


Figure 6.2 Select ship layouts or import a user defined drawing file.

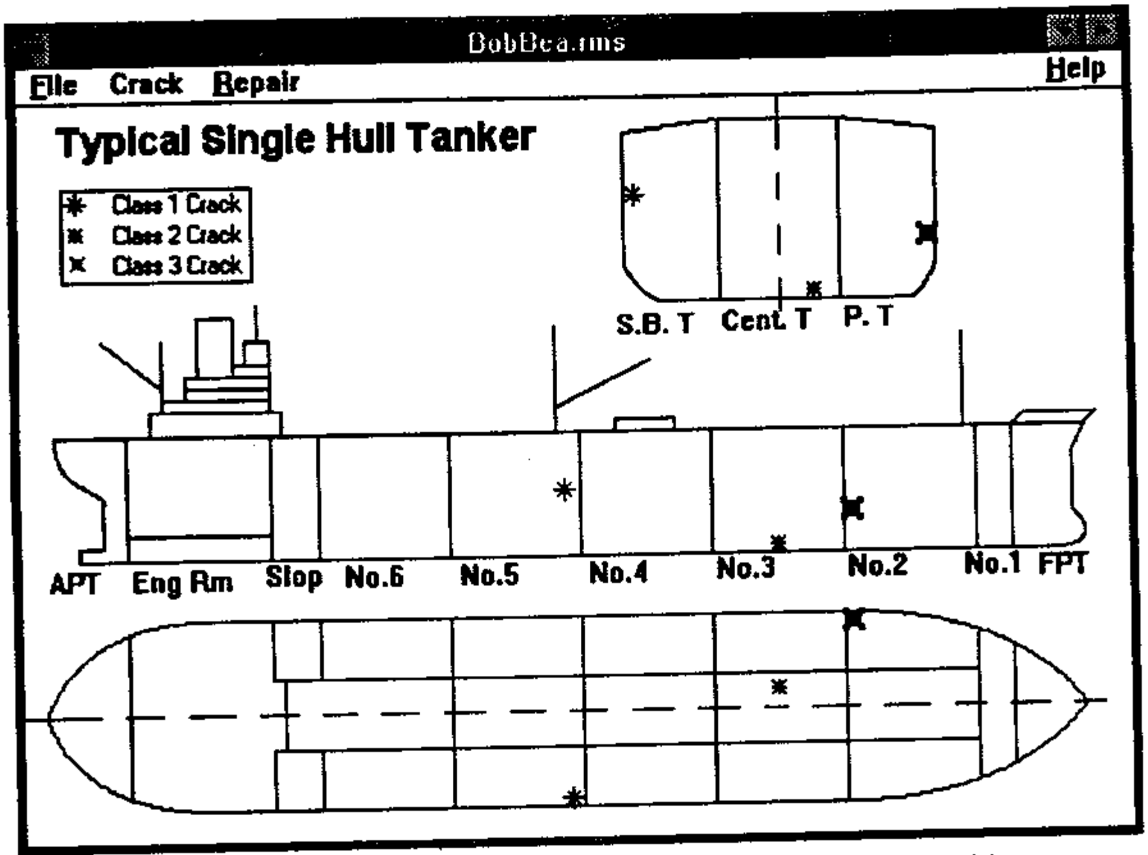


Figure 6.3: Three cracks have been inputted in this general layout.

The three view ship layout successfully solves the difficulty of describing the spatial location within a hull structure of a particular survey result (See Figure 6.3). The graphics tell users the coordinate system quite clearly. And the precise location can be emphasized again by the character based crack data which is inputted by users.

The RMS program uses three view ship layout to locate a crack. This means that users have to input three crack marks to identify one single crack. However if users prefer a two-view layout or a one-view layout (For creating a user customized bitmap, refer to the companion report - User Menu.), they can do that, too. By doing this users have to choose 'Crack-Edit' right after two crack marks are inputted. Figure 6.4 shows an example in that a two-view layout is used.

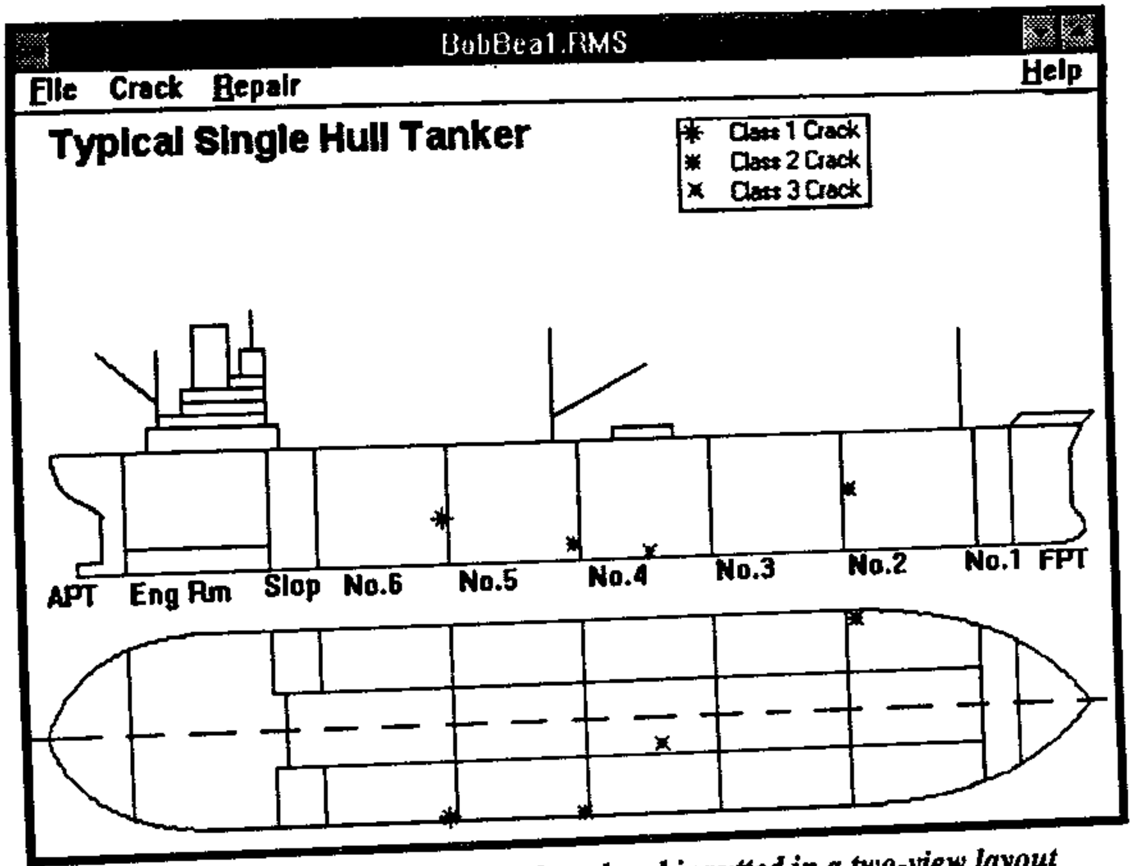


Figure 6.4: Four cracks found and inputted in a two-view layout

After selecting or import a ship layout, a dialogue box can be popped up and allows users to input or edit the general information of the ship as Figure 6.5. There are 8 input fields: ship name, vessel class, owner name, classification, builder name, delivery date, service route and additional information. Each of these fields can store 25 characters.

6.2.3 Repair Databases

No significant repair databases have been developed. However, a catalogue of structural detail failures and suggested repairs was developed and incorporated in the "Guidance Manual for the Inspection and Condition Assessment of Tanker Structures" [TSCF, 1986]. The catalogue has 210 sketches that illustrate the failed details and the proposed repairs. Most sketches show only fractures. Some buckling failures are also included. On each sketch, a list of factors contributing to the failure is described. Some sketches also include repair notes to provide more detailed recommendations, alternative repair methods where appropriate, unsuccessful repairs, and implications for new designs.

Many ship owners and operators have very informal documentation systems for tracking the details of maintenance of a given ship. Documentation ranges from a coherent history of reasonably detailed shipyard repair reports on crack repairs, steel renewals, and coating maintenance to scattered shipyard invoices that define gross tonnage and areas. The documentation varies widely as a function of the diligence of the owner and operator, and as a function of the ship's life.

6.2.4 Critical Area Inspection Plan (CAIP)

Since the report of the Trans-Alaska Pipeline Service (TAPS) Tanker Structural Failure Study found that TAPS tankers experience a disproportionately high number of structural fractures compared to vessels in other trades, these vessels are required to have a Critical Area Inspection Plan (CAIP) by U. S. Coast Guard. CAIP is intended to be the method used by vessel companies to document and track structural failures [USCG, 1990]. In this capacity, CAIP will assist surveyors, inspectors and the vessel's crew to ensure the vessel is properly inspected and maintained. Within the CAIP, the surveyors, inspectors, and crews will be able to find detailed information on the vessel's fracture history, corrosion control systems and previous repairs. The CAIP will also contain a record and evaluation of repairs to the vessel's fractures. It is critical, for any vessel, to known what temporary or permanent repairs have been successful in the past. Repairs completed previously that demonstrate recurring incidence of fractures should not be reused. Furthermore, the evaluation of permanent fixes will be important to the vessel's overall fitness.

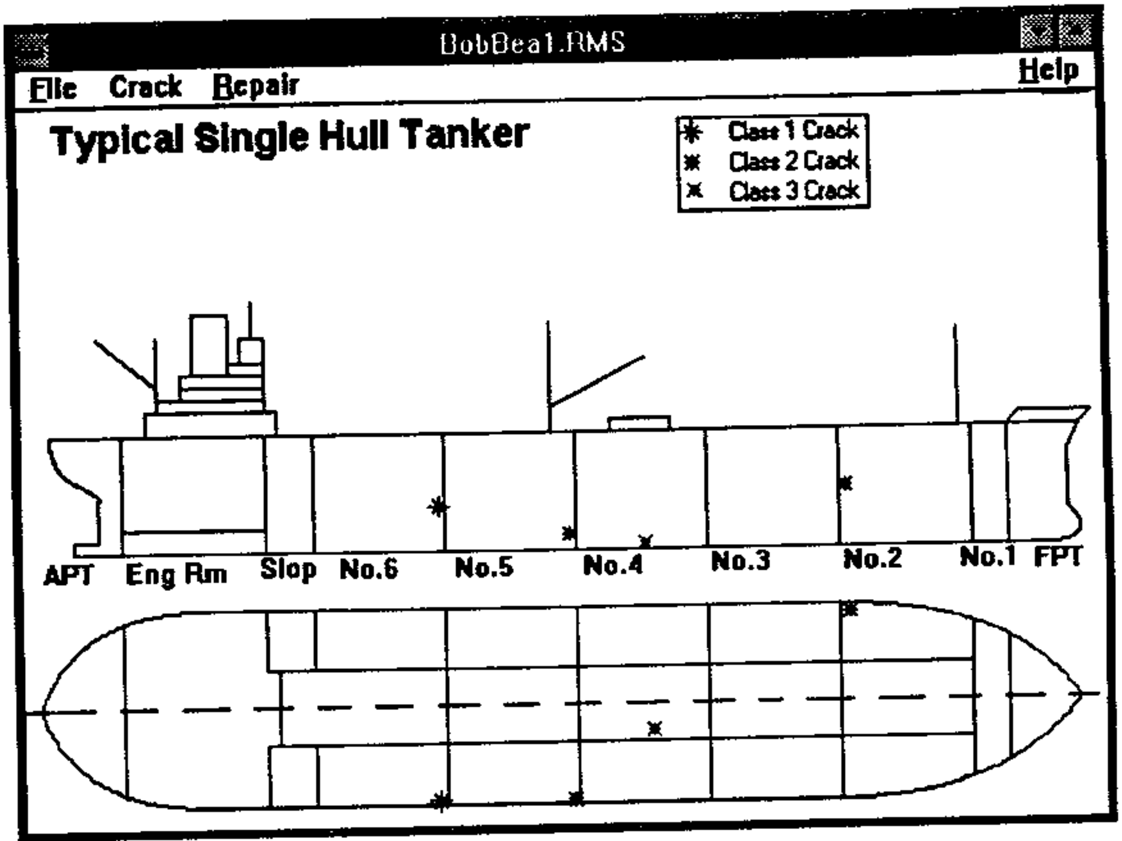


Figure 6.4: Four cracks found and inputted in a two-view layout

After selecting or import a ship layout, a dialogue box can be popped up and allows users to input or edit the general information of the ship as Figure 6.5. There are 8 input fields: ship name, vessel class, owner name, classification, builder name, delivery date, service route and additional information. Each of these fields can store 25 characters.

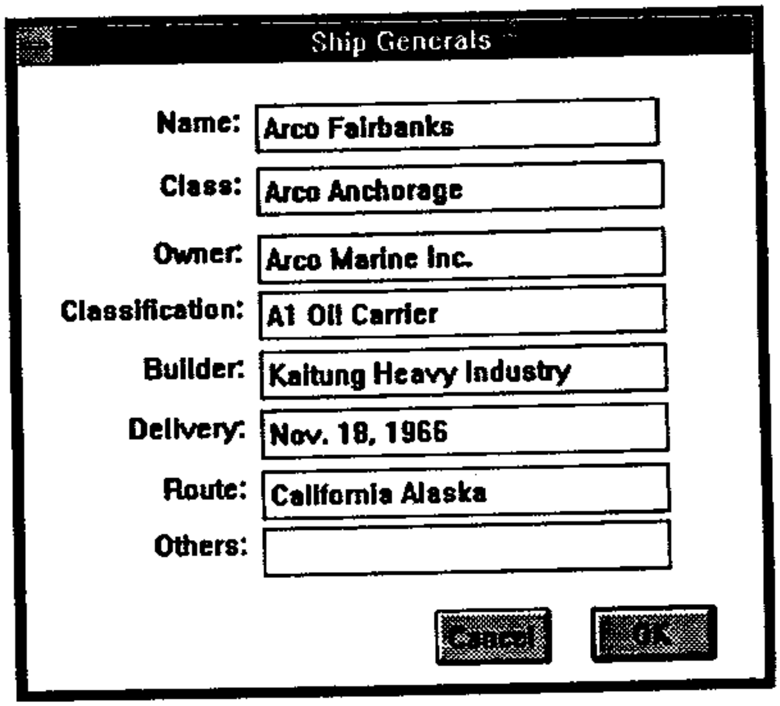


Figure 6.5: Input ship general information.

6.3.2 Crack Data

According U.S. Coast Guard's regulations ship structural failure can be classified into three classes depending on the size of failure and the potential danger. Therefore the RMS program use different colors and the size of crack mark to distinguish them. A large red star sign is assigned to indicate a Class 1 crack. A blue star sign is for Class 2, and a green one for Class 3. The definitions of the three classes are listed as follows:

> **Class 1 Structural Failure:** During normal operating conditions, either (1) a fracture of the oil/watertight envelope that is visible and of any length, or a buckle, that has either initiated in or has propagated into the oil/watertight envelope of a vessel, or (2) a fracture 10 feet or longer in length that has either initiated in or has propagated into an internal strength member.
>
> **Class 2 Structural Failure:** A fracture less than 10 feet in length, or a buckle, that has either initiated in or has propagated into an internal strength member during normal operating conditions.
>
> **Class 3 Structural Failure:** A fracture or buckle that occur under normal operating conditions that does not otherwise meet the definition of either a Class 1 or Class 2 structural failure.

For each crack there are five fields to be input: crack location, finding date, length, repair status and comments. All the fields are character based, so users do not need to memorize any keywords and can simply type in text. The location field can hold 24 characters, and the finding date for 12, the length for 10, the repair status for 25 and the comments for 25 characters. Users can also attach a graphic of a corresponding cracked structural detail to the crack data (see the following two examples in Figure 6.6 and 6.7).

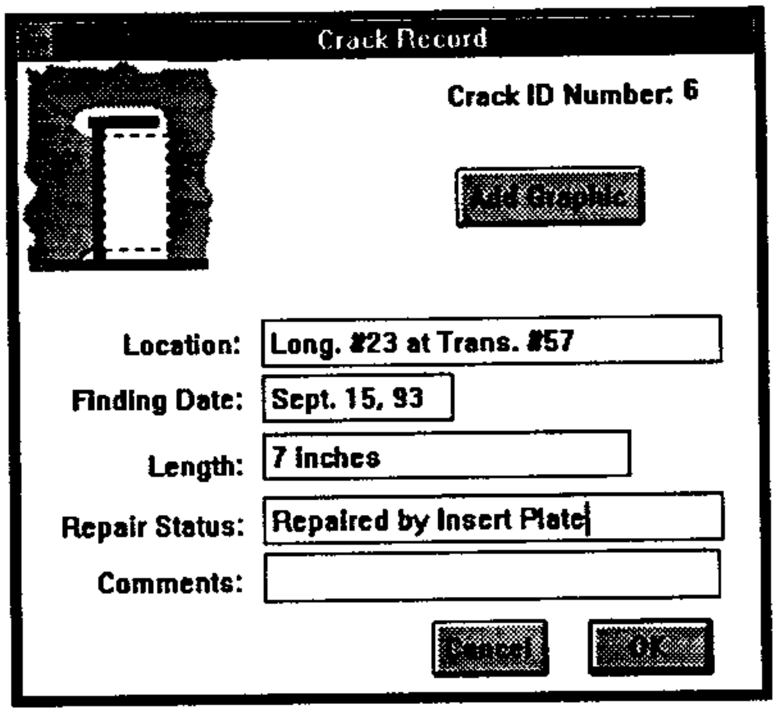


Figure 6.6: This shows an example of new crack data with an attached graphic.

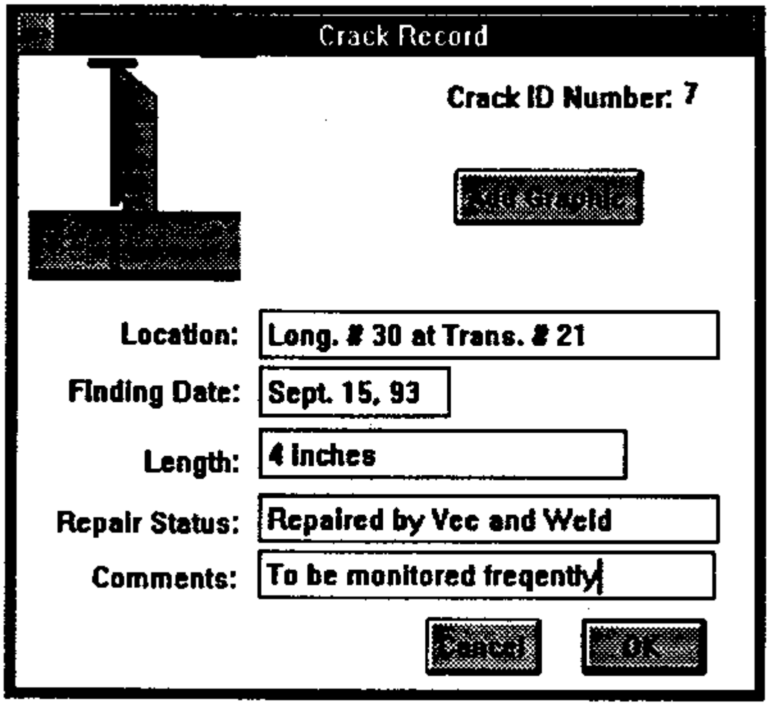


Figure 6.7:* *This shows an another example of new crack data with an attached graphic.

A library of 13 cracked structural details has been created to help users select a graphic easily (see the following figure). A user input mode is also provided to allow users import their own structural detail drawings.

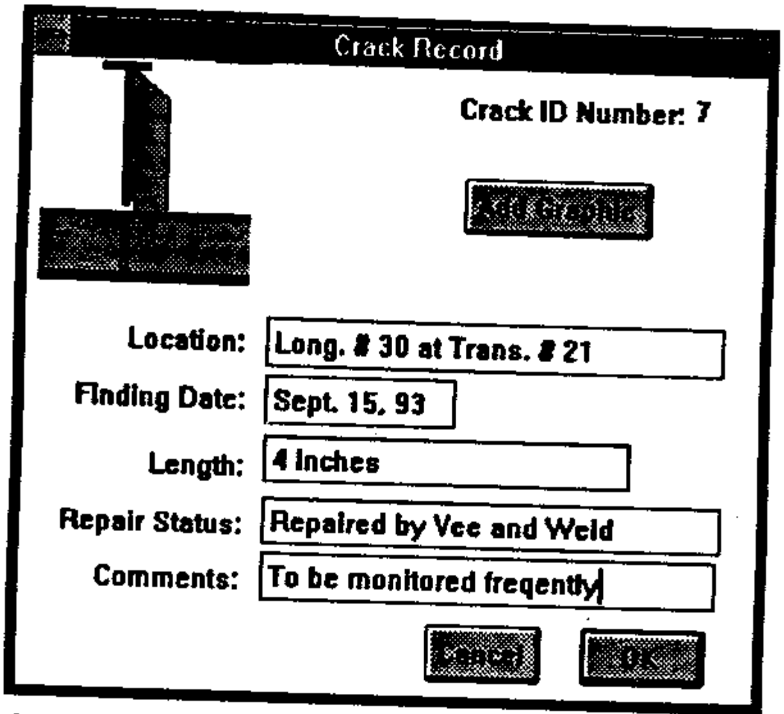


***Figure 6.7:** This shows an another example of new crack data with an attached graphic.*

A library of 13 cracked structural details has been created to help users select a graphic easily (see the following figure). A user input mode is also provided to allow users import their own structural detail drawings.

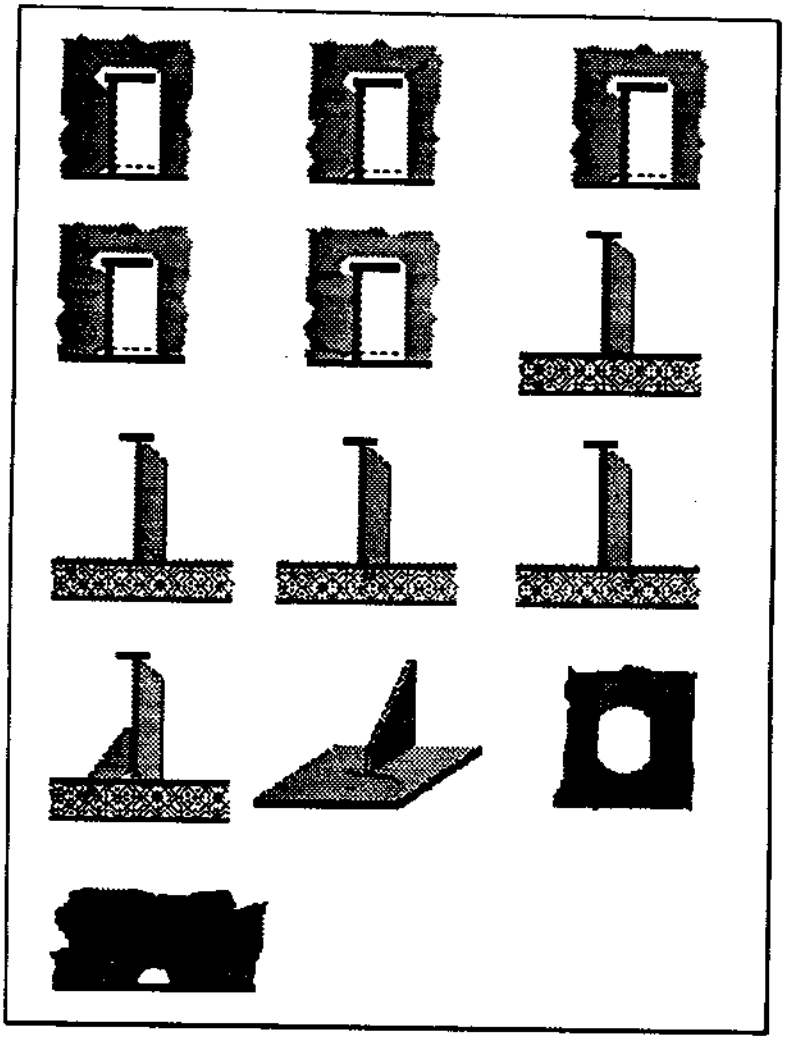


Figure 6.8: There are 13 types of pre-defined cracked structural details.

In summary, RMS crack database has the following advantages:

1. It is one of the most user friendly database which has ever been created. All the operations follows the standards of MS-Windows environment.
2. Inspectors can print out the ship layout that is pre-defined in the RMS as a draft paper before going into tanks. Also they can review the previous data of RMS database to locate the critical area with high likelihood of failures. The RMS simplifies the final inspection report. The final report consists of a floppy disk containing the crack information and the comments regarding the vessel inspection.

4. The RMS has the ability to analyze and evaluate the best repair from a group of repair alternatives.
5. It enhances the efficiency and quality of the inspection and repair. The inspection team and the repair team can both communicate with the home office naval architect, transmitting copies of the information contained on the floppy disks via satellite communications. Naval architects in the home office can then participate in decisions to modify the inspection program or to change the repair specification.
6. The RMS uses a three-view ship layout, a character based description of crack location and a library of cracked detail drawings to specify a particular crack. It is easy to use and understand. Most importantly, it is easy to create a ship layout or a structural detail drawing. Some other database systems which use CAD to locate a crack or other failures may be difficult to operate. In addition, creating a CAD ship model takes a lot of effort, time and money. Other databases that uses keywords to specify a failure location without graphical operating environment are most difficult to use.

CHAPTER 7. RMS CODE AND VERIFICATION

The Repair Management System version 2.0 (RMS 2.0) has been programmed in C to demonstrate the feasibility of the concepts discussed. Due to the limited time available, RMS 2.0 has been developed into a prototype that provide only necessary functions. For a more powerful application, the RMS 2.0 may need more coding effort to enhance the current version of RMS. For information on how to use the RMS 2.0 and how to use or improve the source code, please refer to the companion report, RMS User Menu. In this chapter, a summary of the program and its assumptions is presented followed by a verification of the code.

7.1 Summary of RMS Program

The Code sub directory contains the following files:

Repair.prj	Project file.
Repair.def	Define program environment.
Repair.res	Supplies bitmaps, menu, dialogue boxes, cursor and other resources.
Repair.h	Define public structures and associated constants.
Main.c	Does initialization and created the Main Window.
MainWnd.c	Processes window messages.
FileCmds.c	Performs File Commands for the top Menu Bar.
FileFmt.c	Writes the different file formats.
FileUtil.c	Provides common procedures for file commands.
Analysis.c	Shows data input windows for fatigue life prediction.
CalcuFat.c	Calculates fatigue life.
AddRecrd.c	Processes crack record input dialogue.

The following illustration shows the calling relationship of most procedures.

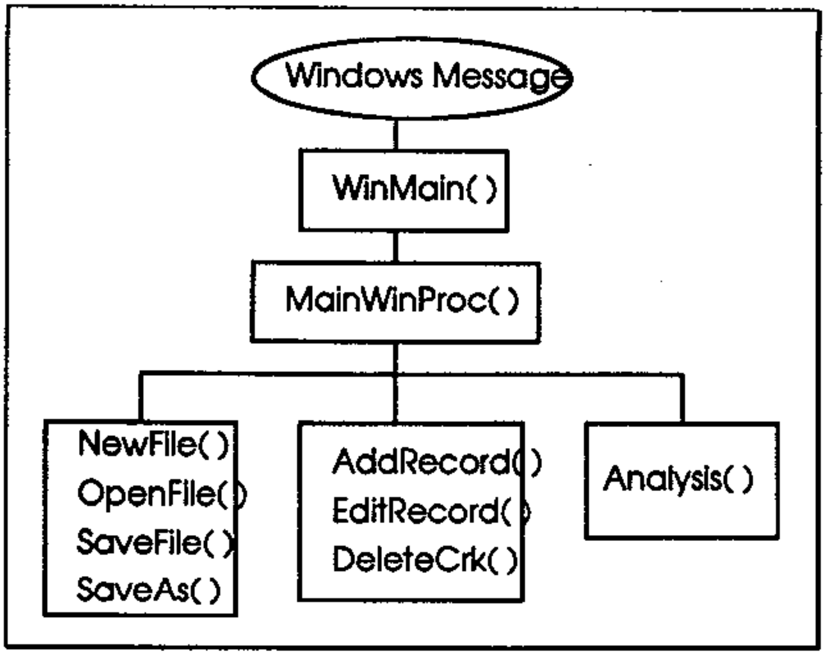


Figure 7.1: This shows the message flow of the RMS program.

A complete listing of the C source code is provided in Appendix A of the companion user menu. The program performs portions of the RMS modules discussed in Chapter 3. The contents of the C code are discussed below in terms of each RMS module.

7.1.1 Windows Module

This module includes Main.c and MainWnd.c. It does initialization and created the main window with a menu bar. It processes window messages like mouse moving, resizing windows, user selecting a command, input form keyboard and others.

7.1.2 File Input Output Module

This module includes FileCmds.c, FileFmt.c, and FileUtil.c. These files provide functions that can input a text file like *.rms and also there are functions for importing bitmap files.

7.1.3 Crack Management Database Module

This module mainly includes AddRecrd.c. It let users to add, delete or edit a record for a particular crack in the graphical ship layout.

7.1.4 Failure Diagnosis Module

No failure diagnosis is conducted. The program assumes the mode of failure is fatigue and the cause of failure is not due to poor quality control at initial construction or due to corrosive effects.

7.1.5 Repair Alternatives Selection (Analysis) Module

This module is the code file, Analysis.c. Detail configurations for any component group (e.g., side shell components) are built into a dialogue box in the program. The graphical detail type selection dialogue box allows users to select different detail types (e.g., longitudinal cutout, flatbar, bracket) and the modified design of each structural details. When redesigning the detail, the original crack location may be either welded or replaced. Since the mode of failure is fatigue, only the crack repair options are considered. These options include vee and weld, vee and weld plus post-weld improvement, add insert plate, and redesign of the detail. The desired repair option can then be selected by the user. In the case of redesign, the user selects from a list of valid detail configurations.

7.1.6 Fatigue Analysis Module

This module is basically the code file, CalcuFat.c. The necessary information to conduct the repair analysis is provided by interactive input from the user and pre-defined data in the program. Ship loading information, including the Weibull parameter, average stress frequency, and expert load zones and ratios are pre-defined in the program. Stress concentration factors for each loading direction and each configuration location, and SN class designations for each location are pre-defined, too. Interactive inputs includes the ship location, detail configuration and failure location, the mean time to failure of the original detail and the desired repair option. With all the information above, the program calculate the expected repair life by using Wirsching equation [Wirsching, 1987]. Repair analysis is conducted only at the location of failure.

7.1.7 Help Module

This module is in the code file, MainWnd.c, along with help script files, Repair1.hlp and HelpHow.hlp. It performs the commands 'How to Use RMS' and 'Repair Information' under the menu bar in the window. The former instructs users how to use all the command in the RMS window. The latter provide users a general introduction on ship maintenance and repair.

7.2 Verification

To demonstrate and verify the code, the RMS is applied to a small side shell structure case study. The repair of a crack in the longitudinal cutout shown in the following figure is explored. Assuming this crack is found while the ship is 10 years old, that means the time needed for this critical spot to crack and grow to the current particular length is about 10 years. The ship owner plans to operate this ship for another 15 years, and wonders what types of repair are available and which one can survive for 15 more years without re-cracking again.

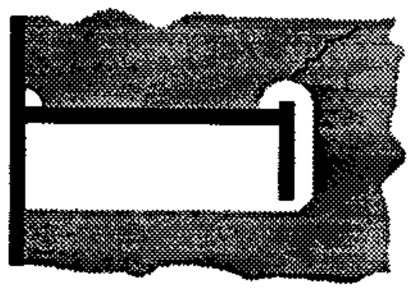


Figure 7.2: A crack is found around longitudinal cutout.

The solution to the ship owner's question is to use RMS. After activate the RMS 2.0. Select Repair-Analysis under the menu bar, and input the cracked detail as following figure.

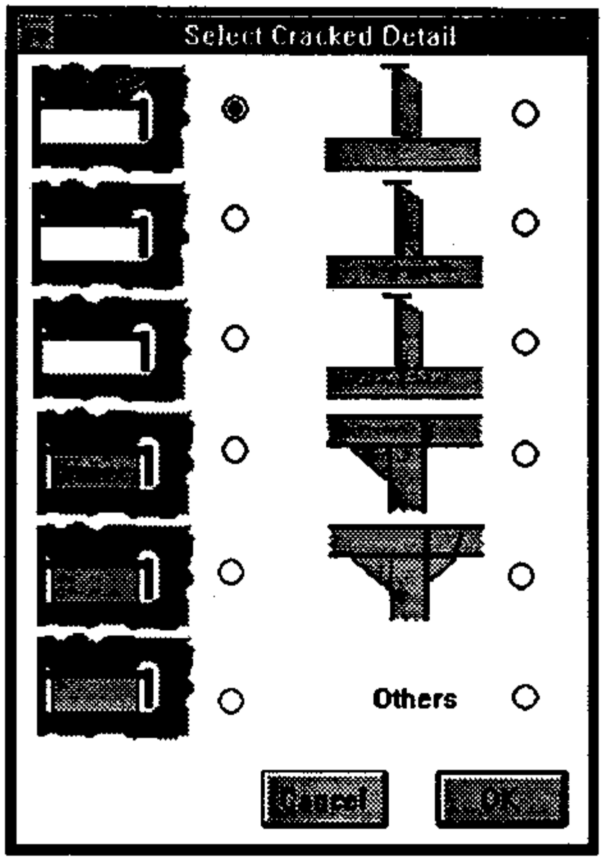


Figure 7.3: Specify the crack spot.

Now another dialogue box will pop up to let you enter the failure time. Since the crack takes 10 years to grow up, let's input '10' here and press OK.

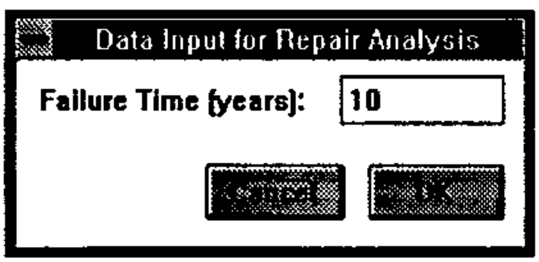


Figure 7.4: Input failure time after selecting 'Analysis'.

Now six repair alternatives are introduced. The corresponding repair lives have been calculated, too. The result shows the vee and weld can last about only 3.0 years. These results match the experience of ship structural repairs quite well. Repair by veeing and welding usually fails again very soon. Apparently this repair is not robust enough to survive the rest 10 years of the ship life in this example. The second repair 'Weld Plus

Postweld Improvement' will last about 4 years. The third repair, inserting a new plate, is something like re-running the fatigue damage cumulation from the starting point of the structure life. It is reasonable to take another 10 years to re-crack and grow to the same length. This repair may not provide sufficient repair life. The rest three repairs are extremely robust. They have repair lives more than hundreds of years. Therefore the betters repair in this case would be design modifications (any one of the last three alternatives).

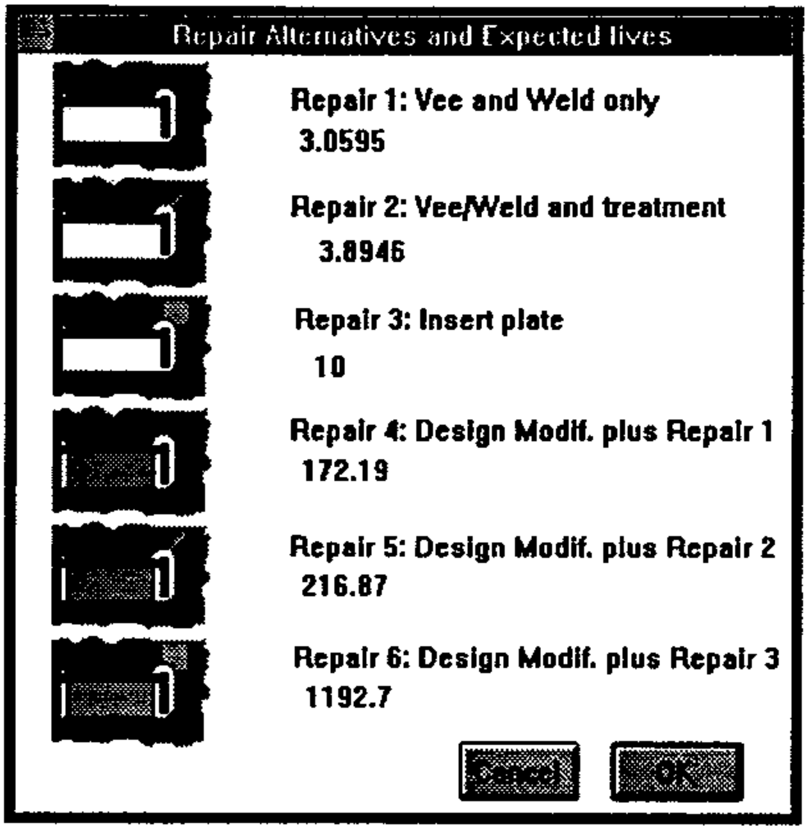


Figure 7.5: The results of estimated repair lives is showed.

Note the stress reduction factors in the SN data file 'REPAIR.DAT' is defined temporarily by human judgment. To draw more conclusions from this case study, additional work is required. This work includes the development of precise stress concentration factors (or stress reduction factors in another way) . In addition, a review of the relative costs, expected interest rates, and the load ratios is necessary. All these will have a significant impact on the decision. With this information and a large database of available CSD configurations, this simple version of the RMS could be a valuable tool for the assessment of repair options.

CHAPTER 8. CONCLUSIONS AND FUTURE DIRECTIONS

8.1 Conclusions

A framework for the development of the second version of Repair Management System (RMS) to aid in ship structural failure diagnosis and repair evaluation has been developed. The RMS is the first known attempt to handle the complexities of ship structural repair analysis in a framework that provides both elements critical to good repair--quick decisions and thorough evaluations.

The RMS follows the natural steps of repair evaluation and includes repair alternatives selection and repair alternative analysis. Research concentration has been placed on the most troublesome problem in ships today: the fatigue damage of critical structural details. To avoid difficult and time consuming finite element analyses, a simplified repair analysis procedure has been developed to fit into the RMS framework. The second version of the RMS specifically designed for the repair of fatigue damage has been developed using a programming environment, Borland™ C.

This research illustrates that, despite the complexities of the repair decision process, the RMS can assist in making quick, intelligent repair decisions for the repair of ships. The RMS outlined in Chapter 7 can be developed into a powerful tool to aid repair engineers in fatigue repair analysis and corrosion repair arrangement. This development effort must include:

- development of a sophisticated database system to easily manage the input data;
- development and maintenance of a complete library of details that represent both old and current designs;
- structuring the finite element analysis results in the RMS stress concentration factor format for quick repair analysis;
- tuning of the load ratios or the development of a new system to determine relative loads (including the possible use of instrumentation); and

- continued verification of the RMS system.

The case study performed on the repair of a transverse cutout failure on side shell structure using RMS 2.0 clearly illustrates the usefulness of this simple RMS version. The RMS can quickly perform a comparative analysis of repairs, and with proper information on the loadings, critical structural details, and costs, consistent repair decisions can be made quickly. In addition, the case study stressed the significance of understanding the durability of the existing structure in order to make intelligent repair decisions. If the durability of the existing structure is not known to some level of confidence, no repair analysis will be successful.

To implement the complete RMS concept envisioned in Chapter 3, significant effort and a long term commitment are required. This effort would involve all phases of repair analysis and require professional programmers to work with naval architects who are familiar with programming language C and MS-Windows environment or other graphical interfaces. High priority in this effort should be placed on proper knowledge representation in ship structural maintenance and repair.

8.2 Future Directions

The repair of ships was used as a basis to discuss the possible application of computer technology to handle a difficult engineering problem. The scope of the current work was highly constrained and limited due to the time available. As a result, many enhancements to the RMS 2.0 and the current research are possible.

8.2.1 Add program features

One suggested enhancement is the expansion and improvement of the program features. The role of the different type of data in the current RMS is to (1) determine the mean life to failure of specific details within the ship based on the historical database, (2) store information on structural components (stress concentration factors) and loadings (stress ratios, Weibull shape factors) and (3) store default repair options for specific damage situations. By integrating existing ship condition databases and developing new and more accurate stress concentration factors, stress ratios and shape parameters, the power of the RMS could be increased quickly. Once the complete RMS system is implemented, expansion to ship components other than side shell structure could proceed, including deck structure, bottom structure, transverse structure, special structure (knuckle joints, etc.), and any other structure of interest.

A handy feature that can greatly improve the particle use of the RMS is to add the Print function. Due to the limited time available and the amount of work to be done in this project, we didn't have time to develop a Print function for the current RMS. However with a Print function, inspectors can print out a ship layout with all the cracks found previously before a hull inspection. This will give inspectors a clear idea on where the cracks may locate. Also repair engineers can easily figure out the condition of a ship through the printout. In RMS 2.0 users can still print out the whole RMS window by pressing 'Alt' and 'PrintScrn' in the same time. The image will be stored in the Window's Clipboard. Users can use any other drawing program to print it out.

Another feature that can be improved is expanding the Help function to provide a clear explanation facility to teach the users of the RMS about repair analysis. This could be a valuable for training tool for repair personnel. There are two Help file in the current RMS, one is to teach users how to use the RMS 2.0. The other one is to provide general information of ship structural maintenance and repair including graphical repair examples, steel repair, maintenance of corrosion protection system and others. More Help files can easily added within the Help command in the Windows menu bar.

8.2.2 Improve repair life estimation

In the current project there is difficulty in selecting a proper S-N curve for a particular repaired critical structural details (CSD). The S-N classification of CSDs used in the projects is mainly based on human judgment. A certain class of S-N curve is matched to a hot spot of a CSD by comparing the similarity of the hot spot geometry and fatigue specimens. Inaccuracy may be introduced in this matching process. Beside the matching process, there is another factor that will introduce inaccuracy. In most fatigue experiments, like those done by U.K. Department of Energy or Munse, small fatigue specimens are used to set up the S-N relations, see [Munse, 1983]. These S-N relations are used intensely by naval architects to imitate full-scale ship CSDs. In this case inaccuracy is introduced due to the scale factor. Therefore it is our wish to establish a S-N classification of repaired CSD in some other more reliable ways.

A recommended way is to establish S-N classification of a particular repaired CSD by regarding in-service ships as a full scale specimen. The full scale experiment will be carried out by imagining the CSDs in in-service ships as fatigue specimens. With gathered historical repair data, the S-N curves of particular repaired CSD can be determined. While the classification is established, a more reliable repair life estimation can be achieved.

For example, a fracture in the circular corner of a longitudinal cutout (or so called slot) was found while the ship was nine years old. We can see that this fracture took nine years to grow up. The repair was done then by grinding out and re-welding according to the decision of a repair engineer. Unfortunately the repaired fracture re-cracked again in three years. Now the problem is defined as to establish an S-N class for the re-welded circular corner of a longitudinal cutout. We already have enough information and are ready to determine the S-N classification of the repair, veeing and welding.

The S-N class of the circular corner of a longitudinal cutout can be matched by C class, see [Chen, 1992]. Nine-year loads which attack the hotspot of the longitudinal cutout can be estimated by a Weibull distribution. The Weibull distribution has two unknown parameters, Weibull parameter and extreme stress range. Now assume the Weibull parameter is 0.9. The fatigue damage factor for the hotspot in the nine years is one since it failed in nine years. By Miner's rule, we can compute the only unknown, the extreme stress range (See the left side of Figure 8.1).

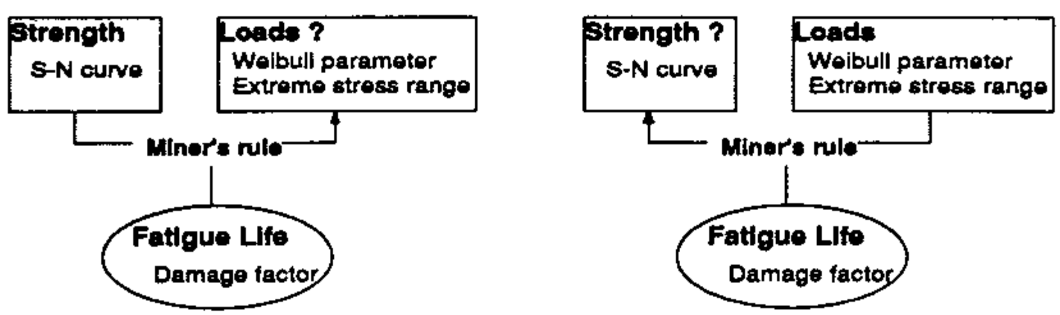


Figure 8.1 This shows the S-N curve calibration process for repaired CSDs.

After the fracture was repaired by veeing and re-welding, the detail re-cracked in three years. With this information, we know that the repair has a fatigue damage factor, one, in the three years. Since the two load parameters, Weibull parameter and extreme stress range, stay the same. Now we can go backward to calculate the S-N parameters slope and intercept value (see the right side of Figure 8.1). Therefore the required S-N class is obtained.

The above introduction presents the procedure of establishing S-N class from one set of inspection data of repair life. While more historical inspection data are gathered, the mean of the estimated S-N parameters can be computed. It will converge to the real mean value while a large number of inspection data are available and applied. The S-N classification of other types of repairs can be developed by the similar procedure.

By incorporating the S-N classification into the RMS, the original goal of developing a more reliable repair life estimating method is accomplished.

8.2.3 Improve crack database utility

The current RMS has a crack management database, users can record cracks on graphical ship layouts. Statistical functions can be added to this database to allow users see the trend or distribution of cracks. Since inspection is such a monumental task on crude oil carriers and other large ships, the RMS could be expanded to guide inspectors to ship locations with the highest probability of failure. This ability would be closely tied to a reliability analysis of the entire ship structure and a tracking of the failure probabilities for all components. Continuous updating of the failure probabilities using historical data or instrumentation is possible. Updated failure probabilities could be used directly for repair analyses.

Another area can be improved is the assessing method of graphical ship layouts. The crack database is organized through a fixed three view ship layout which cannot be zoomed in or out. While a large number of cracks are input, the screen may look a mess. It is possible to rebuild the program such that the fixed layout can be zoomed in and the selected tank drawing will popped up to give users a more detailed look.

8.2.4 Add more failure mode

Failure mode and cause analysis is an obvious area for future improvement, too. A majority of ship failures, especially in crude oil carriers, are clearly due to fatigue. As a result, detailed mode and cause analysis is not currently as important as evaluating fatigue failures. However as ship designs change, new modes and causes of failure may occur. A tool to help evaluate these new modes and causes could prove to be important.

Fatigue crack is not the only mode of failure in ships, but the most common. Other important analyses include buckling, corrosion, global strength, and ship condition assessment. Of these, the ship condition assessment is probably the most important, and more appropriate to the RMS style of analysis. Ship condition assessment is directly related to the ship condition database and could prove invaluable to classification societies in their efforts to keep up with fleets of aging ships.

8.2.5 Other improvement

The important role of instrumentation can be thoroughly evaluated. Much of the discussion in the evaluation of fatigue repair alternatives in the RMS was focused on the estimation of stresses and fatigue damage, and resulted in calculations with high levels of uncertainty. The role of instrumentation would be to reduce the level of uncertainty in order to improve repair and other decisions. Once a good estimate of ship loading patterns is attained through the intelligent use of instruments such as fatigue gauges, strain gauging, accelerometers and others, many exciting avenues of analysis are open. Failure mode and cause evaluation, repair of failures, condition assessment, maintenance predictions, inspection guidance, ballasting and ship operation guidance could all benefit.

REFERENCES

Almar-Naess, A. (ed.), Fatigue Handbook, Offshore Steel Structures, Chapter 6, "Improving the Fatigue Strength of Welded Joints", Tapir, 1985.

Bea, Robert G., "Marine Structural Integrity Programs", Ship Structure Committee Report No. SSC-365, 1992.

Chen, Y.K., "Fatigue Classification of Ship Structural Details", SMP Report No. 1-4, American Bureau of Shipping, August 1992.

DnV, "Inspection, Monitoring, Maintenance/Repair", Report of Committee V.2, 11th International Ship & Offshore Structures Congress, China, 1991.

Gallion, Keith A. and Bea, Robert G., "Repair Management System, A System to Aid in the Diagnosis of Ship Structural Failures and the Evaluation of Repair Alternatives", Structural Maintenance Project for New and Existing Ships, Report No. SMP 4-1, Dept. of Naval Architecture & Offshore Engineering, U.C. Berkeley, June 1992.

ISSC, "Fatigue and Fracture", Proceedings of the Tenth International Ship and Offshore Structures Congress, Volume 1, Lyngby, August, 1988.

ISSC, "Fatigue and Fracture", Proceedings of the Eleventh International Ship and Offshore Structures Congress, 1991.

Jordon, C. R and Cochran, C. S., Newport News Shipbuilding, "In Service Performance of Structural Details", Ship Structural Committee Report No. SSC-272, 1978

Jordon, C. R. and L. T. Knight, "Further Survey of in-Service Performance of Structural Details", Ship Structural Committee Report SSC-294, Newport News Shipbuilding, 1980.

Ma, Kai-tung and Bea, Robert G., "Durability Considerations for New & Existing Ships", Structural Maintenance Project for New and Existing Ships, Report No. SMP 5-1, Dept. of Naval Architecture & Offshore Engineering, U.C. Berkeley, September 1992.

Morrill, J. P. and D. Wright, "A Method for Reasoning By Analogy in Failure Analysis", Transactions of the American Society of Mechanical Engineers, Volume 111, July, 1989.

Morrill, J. P. and D. Wright, "A Model of Categorization for Use in Automated Failure Analysis", Journal of Vibration, Acoustics, Stress, and Reliability in Design, Volume 110/559, October, 1988.

Munse, W.H., Thomas W. Wilbur, Martin L. Tellalian, Kim Nicoll and Kevin Wilson, "Fatigue Characterization of Fabricated Ship Details for Design", Ship Structural Committee Report SSC-318, Department of Civil Engineering, University of Illinois at Urbana-Champaign, 1983.

Roddis, W. M. Kim and Jerome Connor, "Qualitative/Quantitative Reasoning for Fatigue and Fracture in Bridges", Coupling Symbolic and Numerical Computing in Expert Systems, II, J.S. Kowalik and C.T. Kitzmiller (ed.), Elsevier Science Publishers B. V. (North-Holland), 1988.

Schulte-Strathaus Rolf, "Fatigue Database Development and Analysis", Project Report No. SMP 4, Structural Maintenance for New and Existing Ships, Dept. of Naval Architecture & Offshore Engineering, U.C. Berkeley, 1991.

Stambaugh, KA, Wood, WA, "A Non-Expert's Guide for inspecting and Determining the Causes of Significant Ship Fractures" Ship Structure Committee, SSC-337, 1987.

Stambaugh, Karl A. and William A. Wood, "Ship Fracture Mechanisms Investigation", Ship Structural Committee Report SSC-337 (Part 1 and 2), Giannotti and Associates, Inc., 1990

Tikka, Kirsi K., Chevron Shipping, "Inspection and Structural Maintenance of Chevron Double Hull Tankers", 1991.

TSCF, (Tanker Structure Co-operative Forum), "Guidance Manual for the Inspection and Condition Assessment of Tanker Structures", 1986.

TSCT, (Tanker Structure Co-operative Forum), "Condition Evaluation and Maintenance of tanker Structures", 1992.

USCG, "Report on the Trans-Alaska Pipeline Service (TAPS) Tanker Structural Failure Study", Office of Marine Safety Security and Environmental Protection, Washington, D.C, June 25, 1990.

Wirsching, Paul H. and Y.-N. Chen, "Considerations of Probability-Based Fatigue Design for Marine Structures", Society of Naval Architects and Marine Engineers, paper presented to Marine Structural Reliability Symposium, Arlington, Virginia, October 5-6, 1987.

Repair Management System for Critical Structural Details in Ships

User Manual

Kai-tung Ma
and
Professor Robert G. Bea

Department of Naval Architecture and Offshore Engineering
University of California, Berkeley

Table of Contents

Page

111

List of Figures

Page

Repair Management System for Critical Structural Details in Ships

Version 2.0

User Manual

CHAPTER 1. INTRODUCTION

The tasks of building, maintaining, inspecting and repairing very large crude carriers (VLCC) and ultra large crude carriers (ULCC) have become increasingly difficult. These vessels experience varying degrees of internal corrosion and fatigue cracking problems. When a structural failure in the form of cracking or excessive corrosion is discovered by inspection, a decision must be made as to the most effective repair. This decision is not simple because of the vast array of engineering, construction and repair knowledge that must be evaluated. As a result of the complexity and the short time generally available, the proper repair of ships currently relies heavily on the experience of repair engineers. There is simply not enough time to take into account all possible factors and perform detailed analyses. Repair decisions often lack thorough technical and economic evaluation, but serve to get ships back into service quickly.

This poses a key question: **How do we properly manage the inspection and repair data, the existing knowledge of both successful and unsuccessful repairs, the complex analysis tools and additional knowledge to make intelligent and timely repair decisions?** The answer proposed by this research is a Repair Management System (RMS). The RMS is a computerized system to help repair engineers make good repair decisions by integrating a graphical fracture database, structural failure diagnosis and repair alternative evaluation.

The goals of the RMS are to: (1) provide a consistent and repair strategy; (2) enable to make prompt repair evaluations; (3) increase the level of expertise in the shipyard; (4) promote a sharing of repair information among ship owners, operators and shipyards; and (5) utilize analytical and historical ship data. To reach these goals, the RMS 2.0 which equips a fracture database, failure mode selection, and fatigue analysis function is developed.

This project was sponsored by the following three organizations: Arco Marine Inc., Lisnave Ship Yard, Ship Structure Committee. We would like to express our thanks for their generous support.

CHAPTER 2. INSTALLATION

2.1 Backup Disk

Before any installation begins, it is always a good practice to backup the program diskette in the back of the report. We assume you're already familiar with DOS commands or Windows operation. For example, in DOS you'll need the DISKCOPY command to make backup copies of your program disk.

2.2 System Requirements

To run RMS 2.0, you must have a 386 or 486 based PC with 2MB RAM at least, MS DOS 5.0, and Windows 3.0 or above. A math co-processer chip is recommended for a 386 based PC; 486 based PCs come with one.

2.3 Using INSTALL

RMS 2.0 comes with an semi-automatic installation program called INSTALL. It is a batch file that create a directory "\RMS" in your hard drive C and then copy all the files in the floppy into the directory. Follow the instruction here to setup the program.

To install RMS 2.0, insert the floppy disk into a floppy drive. Enter a DOS shell or exit from windows then path to the floppy drive by typing "B:". (If the floppy is inserted into drive A, you should type "A:" instead of "B:".) Press the key 'Enter'. Type "install" and press 'Enter'. RMS 2.0 should be installing now, and a directory "c:\RMS" will be created in your hard drive. It will take up about 1.2 MB hard disk space. All the files will be copied into the directory. However the installation is not complete yet. We have to specify the program group name, item name, and the path of RMS 2.0 to Windows, so type "WIN" to execute Windows now. To add the program group, select New from File menu in Program Manager. The following window will appear, select Program Group and then OK.

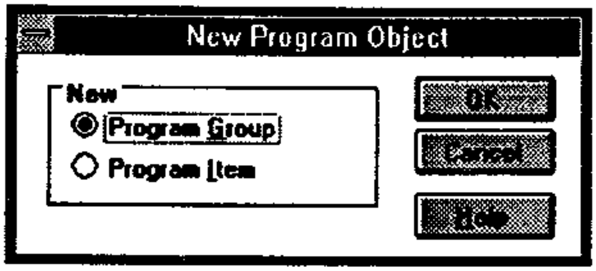


Figure 2.1: Select 'Program Group' for the RMS program.

Next the following window will appear. Fill in the Description and Group File as indicated. Then select OK.

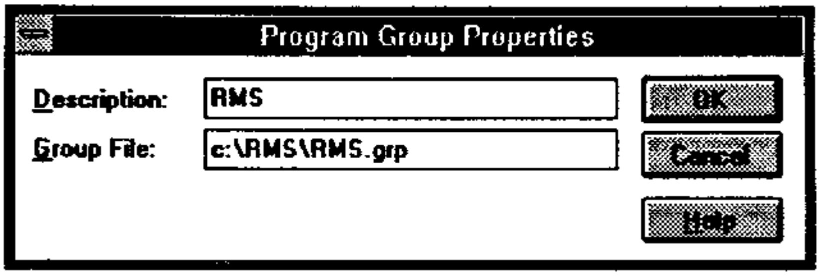


Figure 2.2: Specify the group name and the filename and path.

Notice that a new program group 'RMS' has been created in your Microsoft Windows. Congratulation, you have successfully setup RMS 2.0. Now you can double click the cracked ship icon and are ready to use RMS 2.0.

Figure 2.3: The Program has been installed successfully.

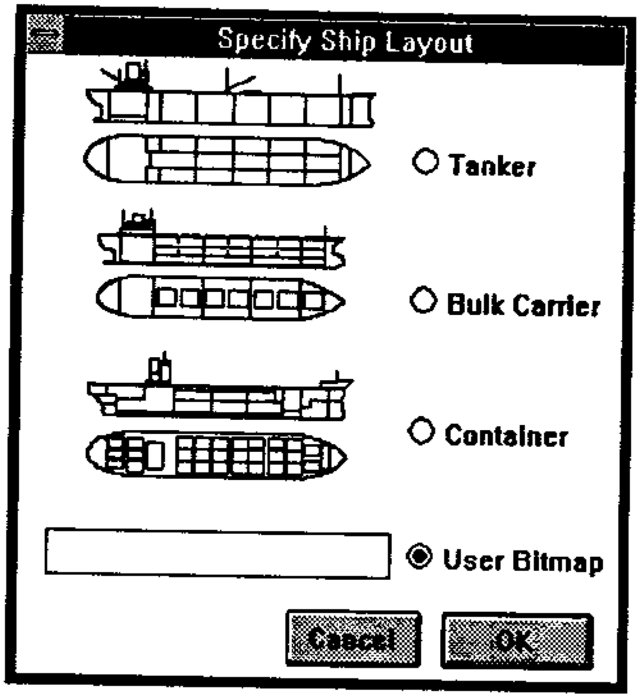


Figure 3.2: This dialogue box pop up after selecting "File-New".

Another dialogue box 'File-New' will pop up. Now input a new filename and push OK button. You can take the ship name as the file name, but be careful that the filename can not be longer than eight letters. Also, you don't have to specify the file extension name. Let's say that we have a tanker named 'BobBea'.

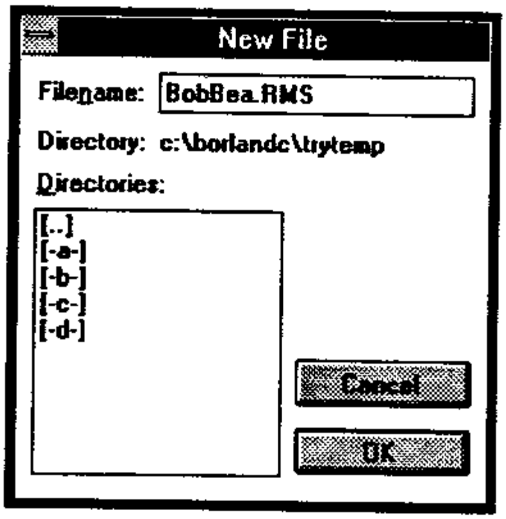


Figure 3.3: Enter a file name.

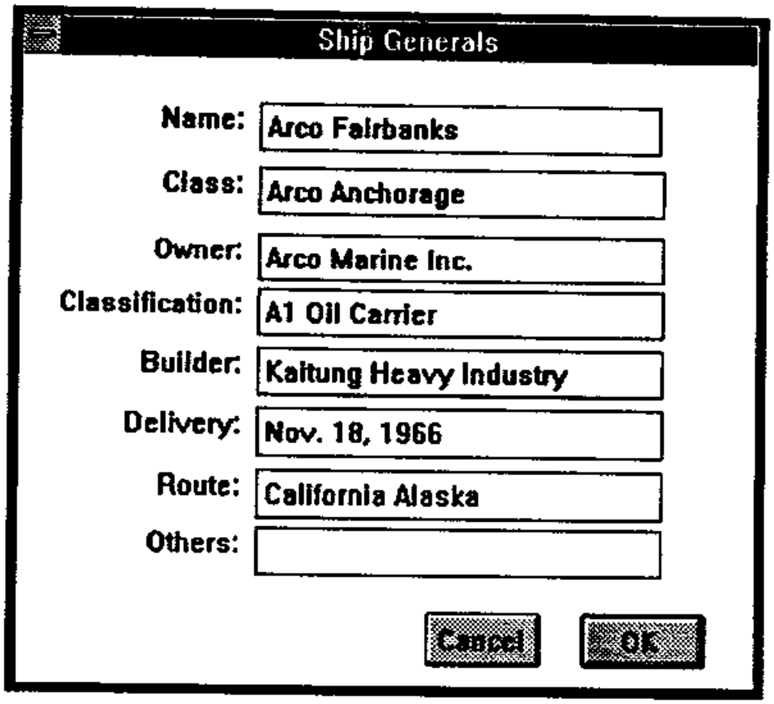


Figure 3.5: Input ship general information.

3.2 Crack Command

Assuming that three cracks are found in the first inspection. One is a 10 feet crack at a side longitudinal cutout in the No. 4 star board tank. Another one is a 5 feet crack at the heel of a flatbar stiffener in the No.3 port tank bottom. The last one is a 1 foot crack in the ladder in the No.5 central tank. According to U.S. Coast Guard's rule these cracks are considered as Class 1, Class 2 and Class 3 crack respectively.

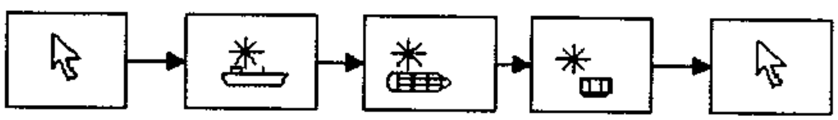


Figure 3.6: The arrow cursor changes into three different cursor while adding a crack.

To input the first crack, select 'Add Class 1' under the Crack command. Notice that the mouse arrow cursor has change into a crack with a ship side view. Move the new cursor to the right location in the side view drawing and click to set the side view crack. Now you should see a red star sign located on the screen. Notice that the cursor has changed into a crack with a ship top view (See the above figure). Set the top view crack on the right location and the cross section view crack in the similar way.

At this moment a crack record input dialogue box will pop up automatically. Input location, finding date, length, ... etc., and press OK. The first set of three red star logo has been placed in the ship drawing. Users can input the other two cracks in the similar way. The second crack (Class 2) should be in blue and the last one (Class 3) in green. Figure 3.4 shows all the three sets of cracks in the ship drawing.

In each crack record there are five fields to be input: crack location, finding date, length, repair status and comments. All the fields are character based, so users do not need to memorize any keywords and can simply type in text. The location field can hold 24 characters, and the finding date for 12, the length for 10, the repair status for 25 and the comments for 25 characters. Users can also attach a graphic of a corresponding cracked structural detail to the crack data (see the following two examples).

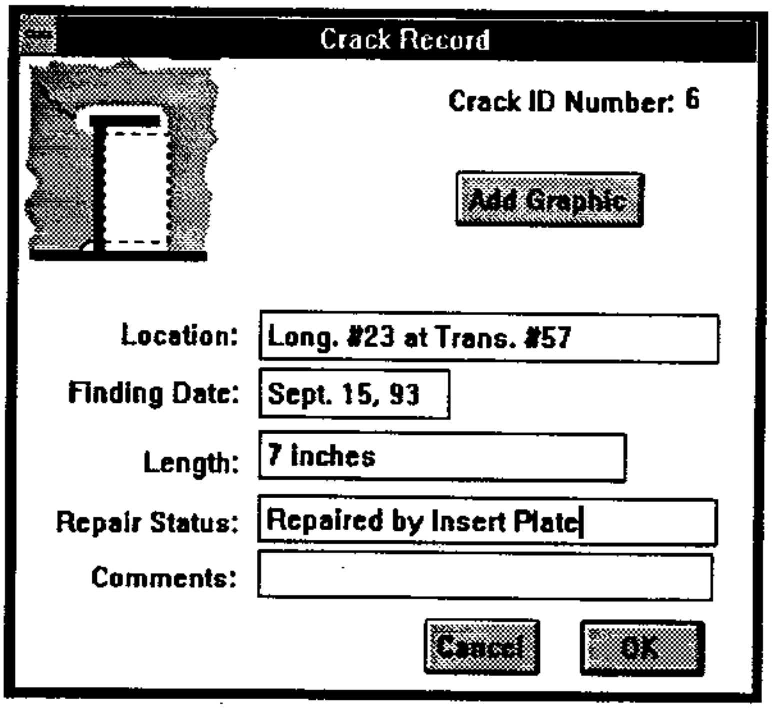


Figure 3.7: This shows an example of new crack data with an attached graphic.

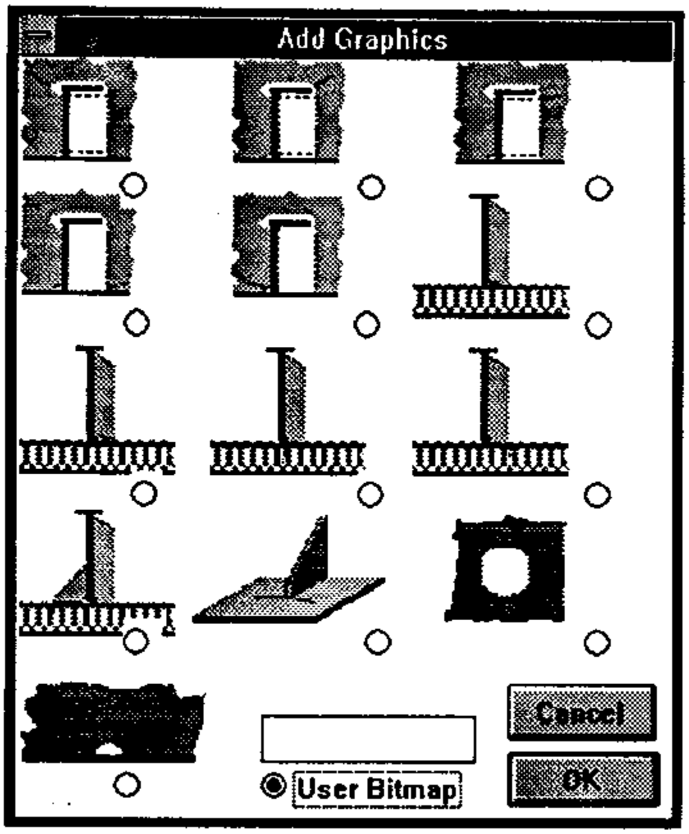


Figure 3.9: There are 13 types of pre-defined cracked structural details.

3.3 Crack Classification

According U.S. Coast Guard's regulations ship structural failure can be classified into three classes depending on the size of failure and the potential danger. Therefore the RMS program use different colors and the size of crack sign to distinguish them. A large red sign is assigned to indicate a Class 1 crack. A blue sign is for Class 2, and a green one for Class 3. The definitions of the three classes are listed as follows:

> **Class 1 Structural Failure:** During normal operating conditions, either (1) a fracture of the oil/watertight envelope that is visible and of any length, or a buckle, that has either initiated in or has propagated into the oil/watertight envelope of a vessel, or (2) a fracture 10 feet or longer in length that has either initiated in or has propagated into an internal strength member.

> **Class 2 Structural Failure:** A fracture less than 10 feet in length, or a buckle, that has either initiated in or has propagated into an internal strength member during normal operating conditions.

Class 3 Structural Failure: A fracture or buckle that occur under normal operating conditions that does not otherwise meet the definition of either a Class 1 or Class 2 structural failure.

3.4 Two View Input

The RMS program uses three view ship layout to locate a crack. This means that users have to input three crack signs to identify one single crack. However if users prefer a two-view layout (For creating a user customized bitmap, refer to the next chapter.), they can input two crack signs, too. By doing this users have to choose 'Crack-Edit' right after two crack signs are inputted. The following figure shows an example in that a two-view layout is used. Users can find the example bitmap file under the RMS directory and is named as "Tanker2V.bmp".

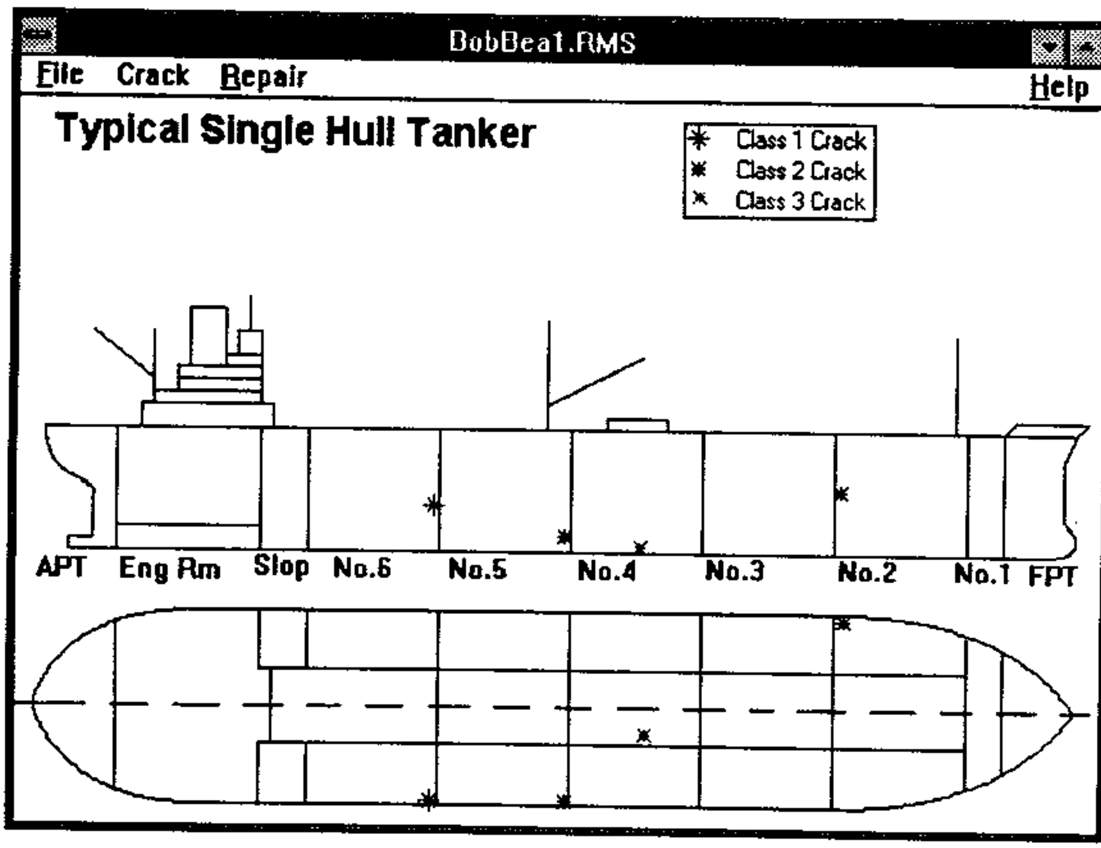


Figure 3.10: Four cracks found and inputted in a two-view layout

If users prefer one-view layout (usually top view), it is also possible to input only one crack sign for one crack. Right after input one sign, users should choose 'Crack-Edit' directly to enter the location, finding date, ... etc.

CHAPTER 3. USING RMS 2.0

After double clicking the RMS icon, the RMS main window will pop up like the following figure. In the main menu some commands may be grayed. They will be black and accessible after a file is opened. The menu bar shows four commands: file, crack, repair and repair. This chapter will introduce the first two. The other two will be reviewed in the later chapters. Those users who are not familiar with windows operation are recommended to follow the step-by-step directions in this chapter.

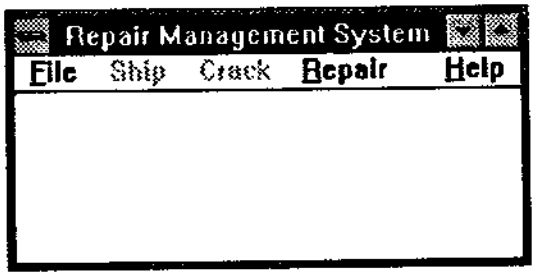


Figure 3.1: RMS 2.0 is popped up.

3.1 File Command

The file command has five functions: new, open, save, save as and exit. Move the mouse cursor to 'File' and click on it. You should see the five functions. Click on the function 'new'. A ship layout selection dialogue box will pop up to let you choose a ship type from a single hull tanker, a bulk carrier, a container or a user defined ship drawing bitmap. The last one will be discussed in details in the next chapter. Let's assume that we are dealing with a single hull tanker, so move mouse cursor to select the radio button of the tanker.

A pre-defined tanker layout show up in the main window. The tanker layout has most typical configuration such that most user can simply adopt it as their single hull tanker's drawing.

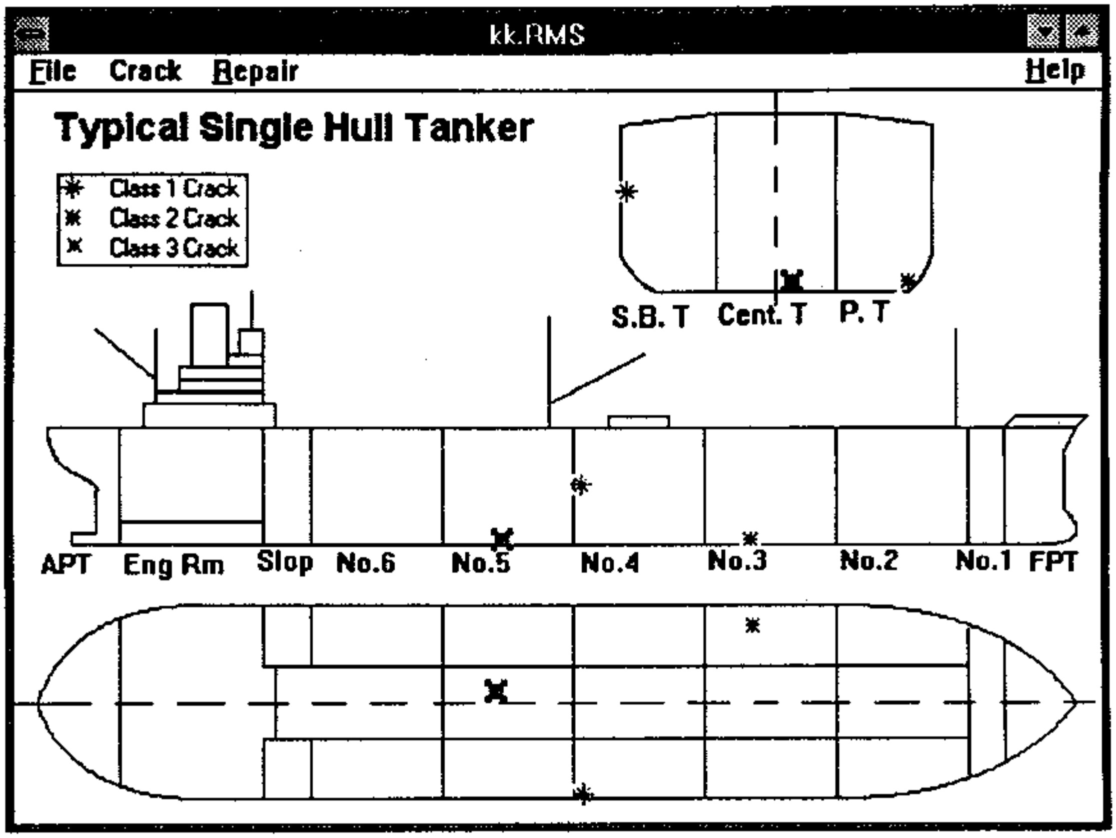


Figure 3.4: Three cracks have been inputted in this general layout.

After selecting or import a ship layout, a dialogue box can be popped up and allows users to input or edit the general information of the ship as the following figure. There are 8 input fields: ship name, vessel class, owner name, classification, builder name, delivery date, service route and additional information. Each of these fields can store 25 characters.

CHAPTER 4. CREATING SHIP LAYOUTS

The RMS 2.0 provide the feature of user-input bitmap by that users can import their own ship drawings into the program. Users can specify a bitmap filename in the dialogue box after selecting FILE_NEW and then import it onto the main window of RMS program. The drawing can be anything such as a oil tanker layout, a bulk carrier layout or even a yacht layout, but they have to be in Window's bitmap formats. The program can handle monochrome (black and white) or 16 (maximum number of color in Window) color bitmaps. Users need to create the bitmaps by using any drawing program in Windows. This chapter will show you how to build a user bitmap drawing.

4.1 Using Scanner

The most convenient way to create a ship layout bitmap might be to use a scanner. You simply scan the ship drawing into a bitmap file and it is done (See the figure below). However it might not be that simple. First you have to find a ship drawing that is on the paper of a right size to ensure a scanner can cover it and also a computer monitor can show the scanned bitmap in whole. Often it is necessary to reduce the size of the source drawing by using a copy machine with the function of scaling. Then you have to scan it carefully into a monochrome or 16 color bitmap. The quality of the scanned result is usually not very good. The lines may looks fuzzy and sometime twisted. You can save the bitmap into the RMS directory if it looks OK. Alternatively you might want to correct and edit the bitmap using Window's drawing programs.

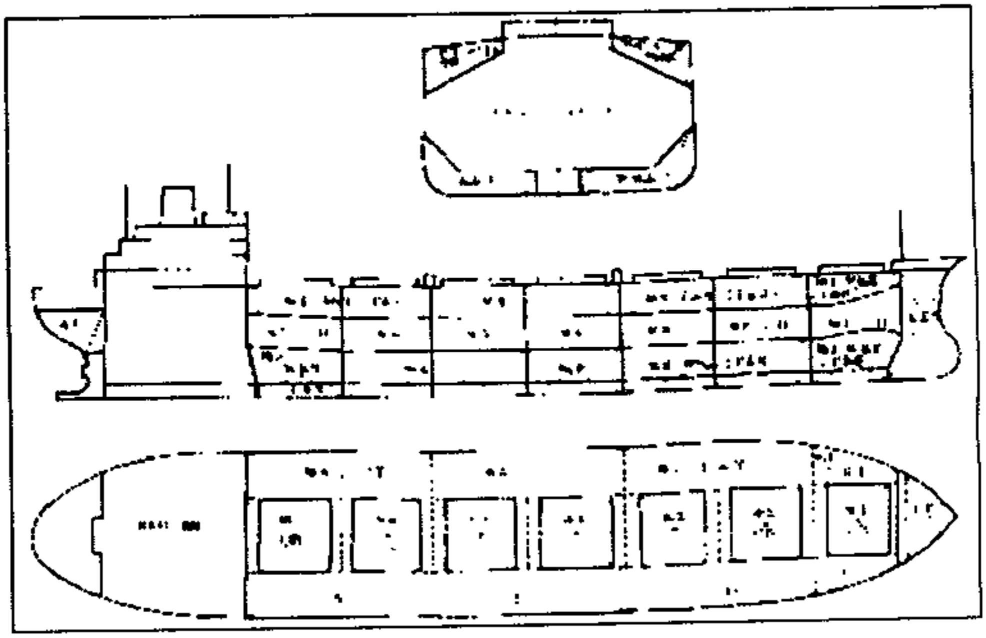


Figure 4.1: A bulk carrier blue print is scanned into this image.

4.2 Using Drawing Programs

The MS Windows itself provide a primitive drawing program call PAINTBRUSH (see the figure below). It can let you perform some basic drawing tasks such as drawing a line, a rectangle, texts ...etc. To learn how to use it please refer to MS Windows menus. However you can purchase more sophisticated drawing programs, too. There are a lot on the market like Corel Draw, Micrographix Designer or other CAD programs. These programs can make the drawing tasks much easier.

Figure 4.2: The drawing program PAINTBRUSH comes with Windows.

The following figure shows the final look of a bulk carrier layout after renovating the scanned image. It looks much better than the original. A title and a legend of three crack classes are pasted onto it. All lines are re-drew. Some labels are attached. This bitmap example can be found in the RMS directory. It is called BULKCARR.BMP. Also another bitmap, LEGEND.BMP, storing the legend of three crack classes is in the directory, too.

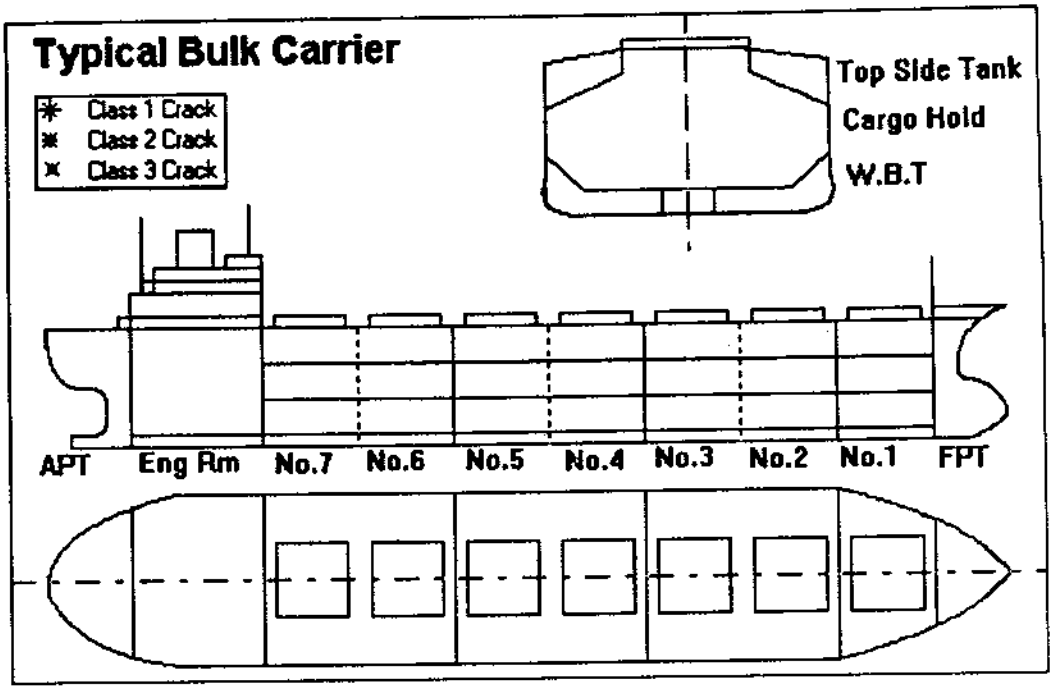


Figure 4.3: This ship drawing is renovated using a Windows drawing program.

You don't necessarily need a scanner to prepare a bitmap. Following the blue print of a ship you can draw the ship layout by yourself using a window drawing program. The layout draw by this way, however, may not show very accurate scantlings.

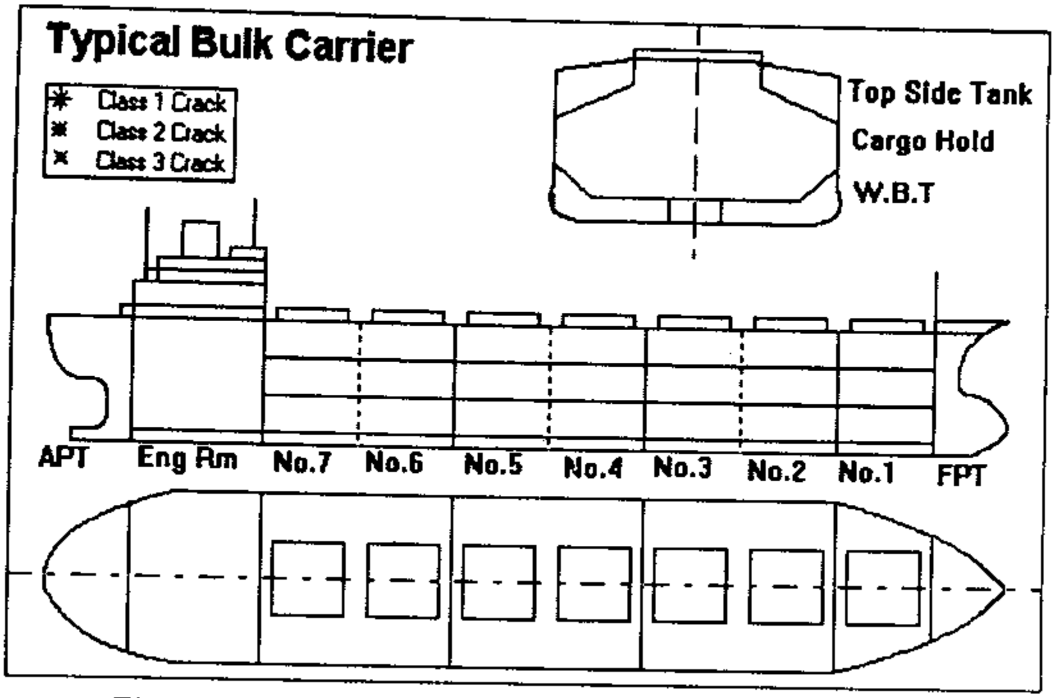


Figure 4.3: This ship drawing is renovated using a Windows drawing program.

You don't necessarily need a scanner to prepare a bitmap. Following the blue print of a ship you can draw the ship layout by yourself using a window drawing program. The layout draw by this way, however, may not show very accurate scantlings.

CHAPTER 5. USING ANALYSIS

The RMS provide a function which can calculate the fatigue life of a certain repair. Users input the time in years in that the critical structural detail develops a failure. Also specify the failed detail, and then the program will calculate and estimate how long the repair can last. This estimated life is quite useful to ship owners. Owners can choose a repair method that is most cost effective and provides sufficient repair life for the continuance of the ship. For more details in the theory of repair life estimation behind this program, refer to the companion theory report.

Let's go through this function by an example. Assume a ship is 10 years old and a crack is found in the longitudinal cutout. The ship owner wonder which repair method should be applied. Available repair methods are welding only, welding plus postweld improving, insert a new plate, modify design by adding a lug, and others. The owner plans to keep the ship for another 10 years. We can help the owner to solve this problem by using the Analysis function in RMS program.

First activate the command 'Analysis' in the pull down menu, and a dialogue box will pop up to let you specify the crack spot. Assuming the crack is in the upper corner of a longitudinal cutout, select the corresponding radio button and press OK.

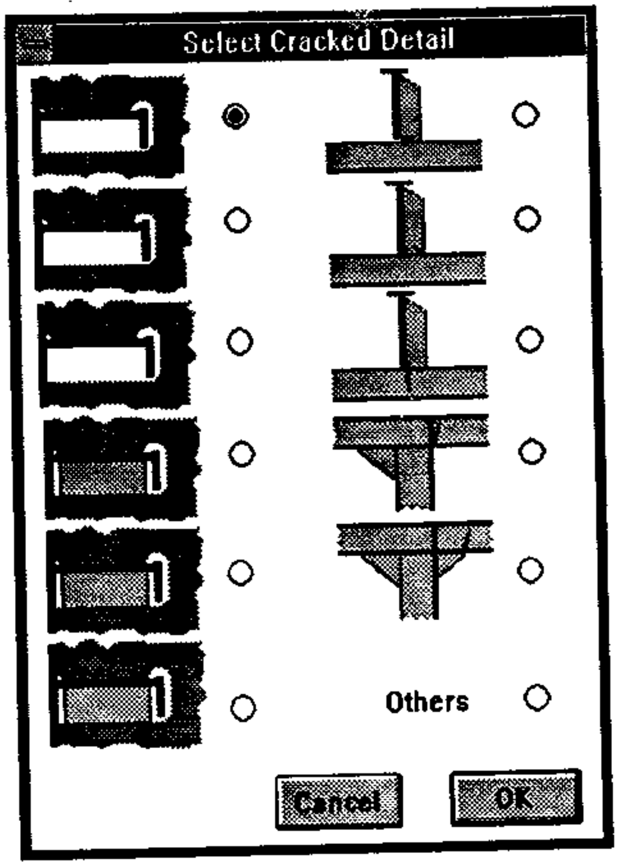


Figure 5.1: Specify the crack spot.

Now another dialogue box will pop up to let you enter the failure time. Since the crack takes 10 years to grow up, let's input '10' here and press OK.

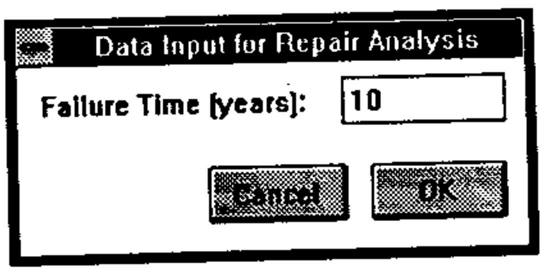


Figure 5.2: Input failure time after selecting 'Analysis'.

Now six repair alternatives are introduced. The corresponding repair lives have been calculated, too. The result shows the vee and weld can last about only 3.0 years. Apparently this repair is not robust enough to survive the rest 10 years ship life.

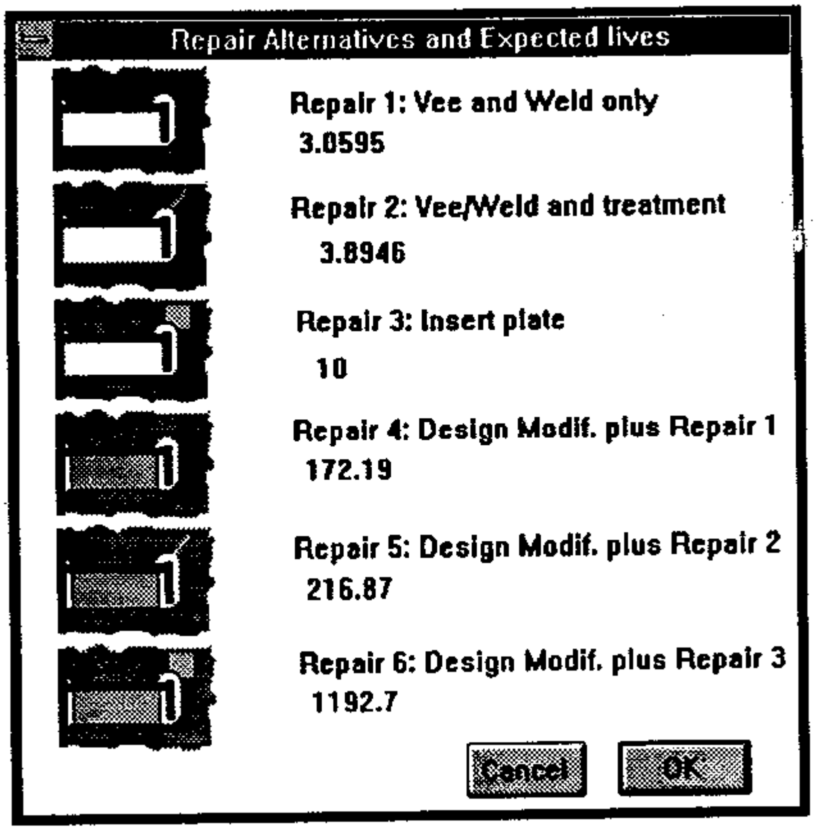


Figure 5.3: The result of estimated repair life is showed.

The second repair 'Weld Plus Postweld Improvement' will last about 4 years and the last one 'Insert New Plate' about 10 years. It means the repaired structural detail will recrack to the same size of this crack at about 20 years old. This repair may be a good choice. The rest three repairs are extremely robust. They have repair lives more than hundreds of years. Therefore the betters repair in this case would be design modifications (the rest three) if we want a more reliable repair.

CHAPTER 6. USING HELP

This chapter teach users how to use the Help which is on the right end of the menu bar. The help menu includes three commands: 'how to use RMS', 'repair information' and 'about'. For users familiar with windows operation 'how to use RMS' and 'about' can be skipped, however 'repair information' may still help them better understand repair methods for different types of failure.

6.1 How to Use RMS

Pull down the help menu, and click on the command 'How to Use RMS'. A help window should then pop up. It contains information on the background of the research project, how to get started, menu bar commands and a U.S. Coast Guard classification of ship structural failure.

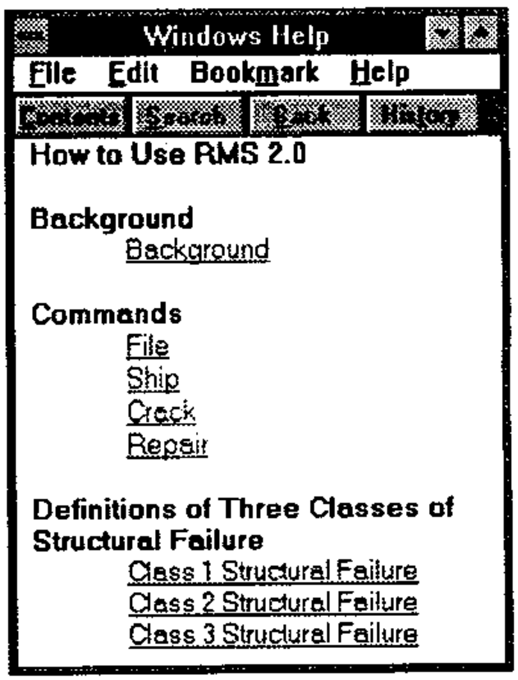


Figure 6.1 The help window on 'How to Use RMS' pop up and the keywords can be clicked to get further information.

By clicking on these keywords which are in green, users can get detailed instruction or explanation on them.

6.2 Repair Information

Repair of failures vary widely. For examples, repairs of cracks can range from temporary cold patches to stop leaks to complete re-design of the structural detail and replacement of steel nearby the detail. It is difficult to decide which repair method is most reliable and cost effective for a particular crack. Experience indicates that many of these repairs must be repeated in subsequent dry docking.

In order to provide RMS users information of historical repair experience, an help file of repair information has been placed under the menu 'help'. While it is selected, a help window should pop up like the following figure.

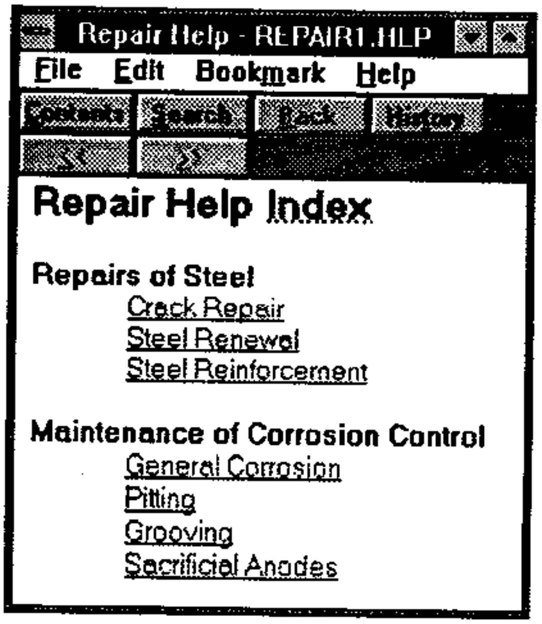


Figure 6.2: ***The repair help window pop up and the keywords can be clicked to get further information.***

Three types of steel repairs are introduced in the repair information help: steel renewal, steel reinforcement, and crack repair. Also there are four types of maintenance of corrosion controls. Users can click those keywords which are in green to get further information. Try click the keyword 'Crack Repair'. The help window should have been renewed like the following figure.

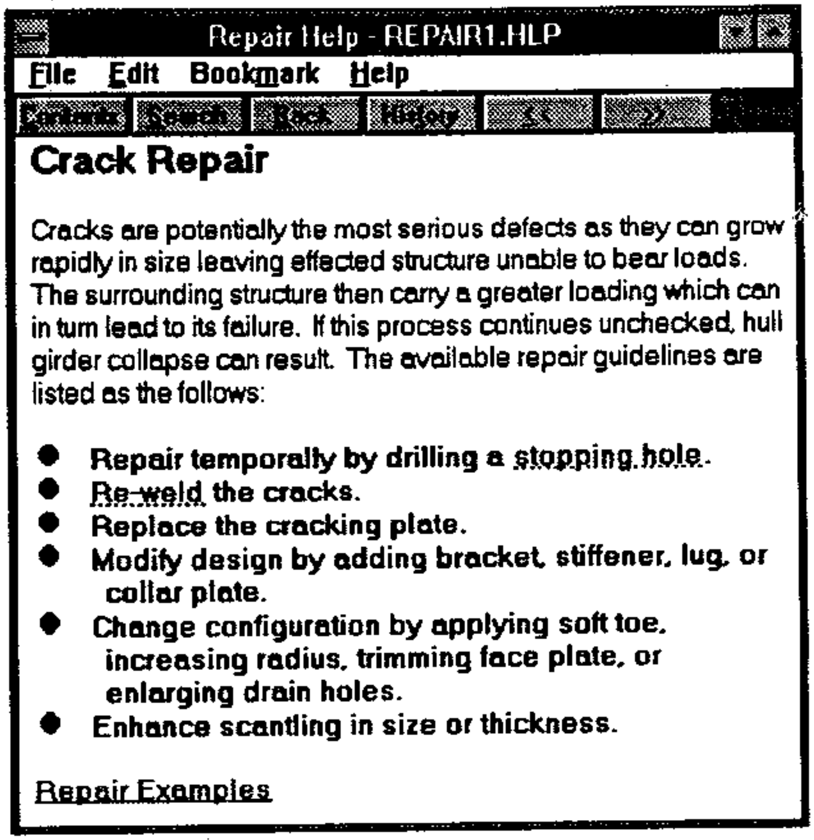


Figure 6.3: The information about crack repair is showed

Now try click on the keyword 'Repair Example'. The titles of fourteen examples should show up. Click on any one of them. Say, try the second one.
A cracked detail should present like the following figure.

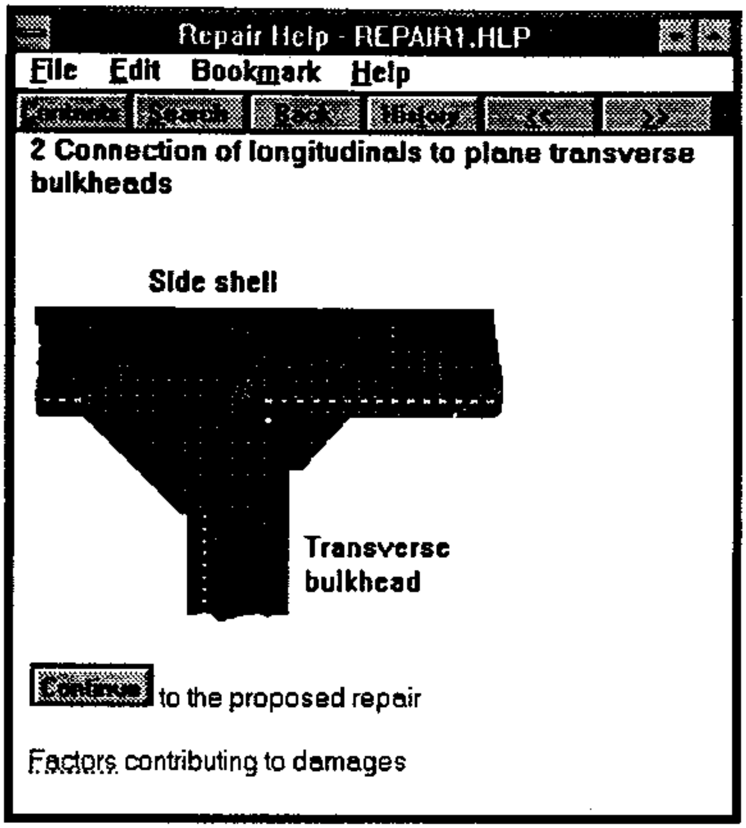


Figure 6.4: This is one of the 14 graphical crack repair examples. A recommended repair method will be showed while the Continue button is pressed.

Try push the Continue button. The recommended repair method is shown as below.

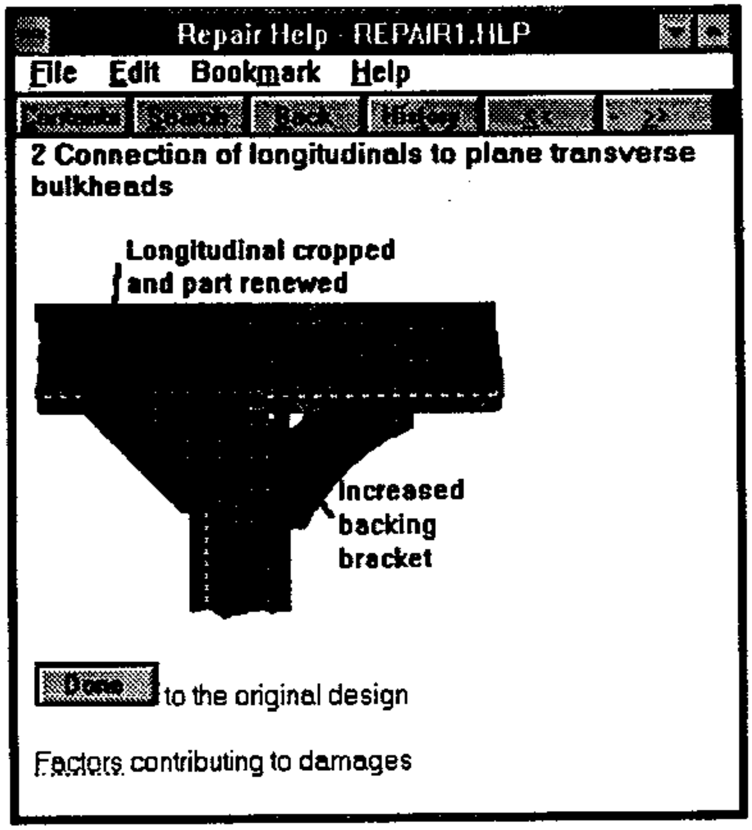


Figure 6.5: The recommended repair method is showed.

The catalogue of 14 structural detail crack failures that has been created in the repair information help is adopt from the book "Guidance Manual for the Inspection and Condition Assessment of Tanker Structures" by the Tanker Structure Cooperative Forum [3]. Information on the experiences of structural detail failures in that book was supplied by the Forum members. Approximately 210 sketches are gathered in it. For more cases other than the 14 examples build in the help file, refer to Reference [3].

CHAPTER 7. USING SOURCE CODE

Due to the limited time available, RMS 2.0 has been developed into a prototype that provide only necessary functions. There is a chance that user may like to add code to enhance the features not provided. For example, user might want to print out the ship layout along with all the inputted cracks, but the current version doesn't provide the print function. In that case, the information contained in this chapter can help users to get started on using the source codes.

7.1 Development Environment

The RMS 2.0 was developed on a 486 based PC with 8 MB RAM and 200 MB hard disk. Borland C++ 3.1 was used for the development of RMS 2.0. The Borland C++ environment requires about 45 MB hard disk. Microsoft DOS 5.0 and Windows 3.1 were used as the Operating System and Graphical User Interface.

7.2 Installing the Source Code

The source code is stored in the program disk under the directory '\code'. It has been archived using PKZIP to compress the data in order to fit on one disk. You should see three files there: pkunzip.exe, readme.txt, and rmscode.zip. The installation batch file only installs the RMS program and data files into your hard disk, and will not install the source code. Therefore you have to manually copy the three files onto your hard disk. And use the decompression program 'PKUNZIP.EXE' to decompress the code files. Simply type 'PKUNZIP RMSCODE.ZIP' to archive these files.

7.3 Overview of Source Files

After the file 'rmscode.zip' is decompressed, you should see the following files:

Repair.prj	Project file.
Repair.def	Define program environment.
Repair.dat	Define SN data and stress reduction factors for all repair alternatives.
Repair.res	Supplies bitmaps, menu, dialogue boxes, cursor and other resources.
Repair.h	Define public structures and associated constants.
Main.c	Does initialization and created the Main Window.

MainWnd.c Processes window messages.
FileCmds.c Performs File Commands for the top Menu Bar.
FileFmt.c Writes the different file formats.
FileUtil.c Provides common procedures for file commands.
Analysis.c Shows data input windows for fatigue life prediction.
CalcuFat.c Calculates fatigue life.
AddRecrd.c Processes crack record input dialogue.

To access the source code, enter Windows and invoke the Borland IDE (integrated development environment). Open project file repair.prj in the code sub-directory. For information on using the IDE see the Borland C++ Users Guide.

7.4 Calling Order of Procedures

The following set of illustrations show the calling relationship of major procedures.

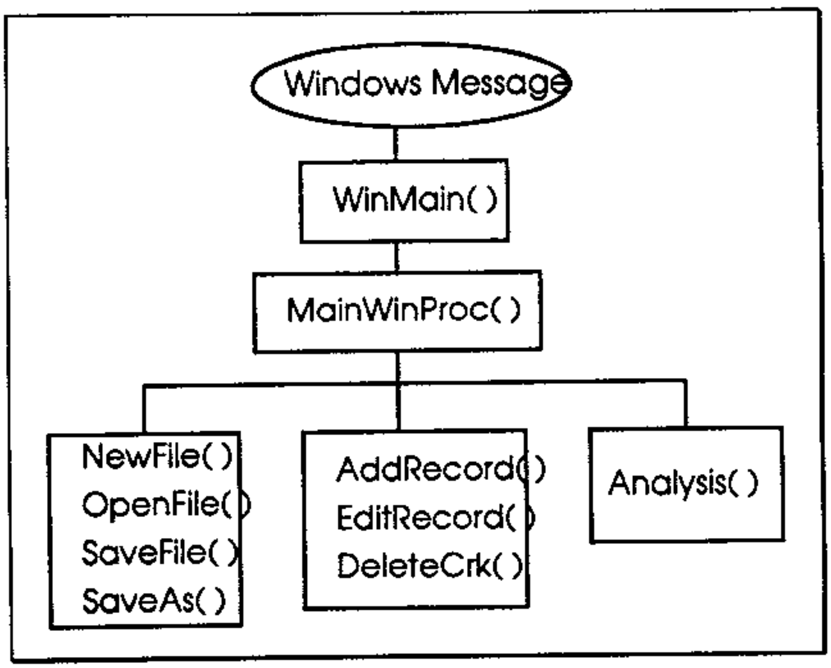


Figure 7.1: This shows the message flow of the RMS program.

REFERENCES

[1] Keith A Gallion, "Repair Management System, A System to Aid in the Diagnosis of Ship Structural Failures and the Evaluation of Repair Alternatives", Structural Maintenance Project for New and Existing Ships, Report No. SMP 4-1, Dept. of Naval Architecture & Offshore Engineering, U.C. Berkeley, 1992.

[2] Kai-tung Ma, "Durability Considerations for New & Existing Ships", Structural Maintenance Project for New and Existing Ships, Report No. SMP 5-1, Dept. of Naval Architecture & Offshore Engineering, U.C. Berkeley, 1992.

[3] Tanker Structure Co-operative Forum, "Guidance Manual for the Inspection and Condition Assessment of Tanker Structures", 1986.

[4] Stambaugh, KA, Wood, WA, "A Non-Expert's Guide for inspecting and Determining the Causes of Significant Ship Fractures" Ship Structure Committee, SSC-337, 1987.

[5] Borland, "Borland C++ 3.1 Application Framework", 1992.

[6] Morrill, J. P. and D. Wright, "A Method for Reasoning By Analogy in Failure Analysis", Transactions of the American Society of Mechanical Engineers, Volume 111, July, 1989.

[7] Roddis, W. M. Kim and Jerome Connor, "Qualitative/Quantitative Reasoning for Fatigue and Fracture in Bridges", Coupling Symbolic and Numerical Computing in Expert Systems, II, J.S. Kowalik and C.T. Kitzmiller (ed.), Elsevier Science Publishers B. V. (North-Holland), 1988.

[8] USCG, "Report on the Trans-Alaska Pipeline Service (TAPS) Tanker Structural Failure Study", Office of Marine Safety Security and Environmental Protection, Washington, D.C, June 25, 1990.

[9] Jordon, C. R and C. S. Cochran, "In Service Performance of Structural Details", Ship Structural Committee Report SSC-272, Newport News Shipbuilding, 1978

[10] Jordon, C. R. and L. T. Knight, "Further Survey of in-Service Performance of Structural Details", Ship Structural Committee Report SSC-294, Newport News Shipbuilding, 1980.

[11] Jordan, CR, Cochran, CS, Newport News Shipbuilding, "In-service Performance of Structural Details", Ship Structure Committee, Report No. SSC-272.1978.

APPENDIX A: SOURCE CODE

A.1 REPAIR.DAT

```
Detail#1Repair#1: 1.08e14 3.5 3.29e12 3.0 1
Detail#1Repair#2: 1.08e14 3.5 3.99e12 3.0 1
Detail#1Repair#3: 1.08e14 3.5 1.08e14 3.5 1
Detail#1Repair#4: 1.08e14 3.5 3.29e12 3.0 0.3333333
Detail#1Repair#5: 1.08e14 3.5 3.99e12 3.0 0.3333333
Detail#1Repair#6: 1.08e14 3.5 1.08e14 3.5 0.3333333

Detail#2Repair#1: 1.08e14 3.5 3.29e12 3.0 1
Detail#2Repair#2: 1.08e14 3.5 3.99e12 3.0 1
Detail#2Repair#3: 1.08e14 3.5 1.08e14 3.5 1
Detail#2Repair#4: 1.08e14 3.5 3.29e12 3.0 0.3333333
Detail#2Repair#5: 1.08e14 3.5 3.99e12 3.0 0.3333333
Detail#2Repair#6: 1.08e14 3.5 1.08e14 3.5 0.3333333

Detail#3Repair#1: 1.23e12 3.0 3.68e11 3.0 1
Detail#3Repair#2: 1.23e12 3.0 5.66e11 3.0 1
Detail#3Repair#3: 1.23e12 3.0 1.23e12 3.0 1
Detail#3Repair#4: 1.23e12 3.0 3.68e11 3.0 0.5
Detail#3Repair#5: 1.23e12 3.0 5.66e11 3.0 0.5
Detail#3Repair#6: 1.23e12 3.0 1.23e12 3.0 0.5

Detail#4Repair#1: 1.08e14 3.5 3.29e12 3.0 1
Detail#4Repair#2: 1.08e14 3.5 3.99e12 3.0 1
Detail#4Repair#3: 1.08e14 3.5 1.08e14 3.5 1

Detail#5Repair#1: 1.08e14 3.5 3.29e12 3.0 1
Detail#5Repair#2: 1.08e14 3.5 3.99e12 3.0 1
Detail#5Repair#3: 1.08e14 3.5 1.08e14 3.5 1

Detail#6Repair#1: 1.23e12 3.0 3.68e11 3.0 1
Detail#6Repair#2: 1.23e12 3.0 5.66e11 3.0 1
Detail#6Repair#3: 1.23e12 3.0 1.23e12 3.0 1

Detail#7Repair#1: 1.23e12 3.0 3.68e11 3.0 1
Detail#7Repair#2: 1.23e12 3.0 5.66e11 3.0 1
Detail#7Repair#3: 1.23e12 3.0 1.23e12 3.0 1
Detail#7Repair#4: 1.23e12 3.0 3.68e11 3.0 0.1
Detail#7Repair#5: 1.23e12 3.0 5.66e11 3.0 0.1
Detail#7Repair#6: 1.23e12 3.0 1.23e12 3.0 0.1

Detail#8Repair#1: 1.23e12 3.0 3.68e11 3.0 1
Detail#8Repair#2: 1.23e12 3.0 5.66e11 3.0 1
Detail#8Repair#3: 1.23e12 3.0 1.23e12 3.0 1
Detail#8Repair#4: 1.23e12 3.0 3.68e11 3.0 0.2
Detail#8Repair#5: 1.23e12 3.0 5.66e11 3.0 0.2
Detail#8Repair#6: 1.23e12 3.0 1.23e12 3.0 0.2

Detail#9Repair#1: 1.73e12 3.0 5.66e11 3.0 1
Detail#9Repair#2: 1.73e12 3.0 1.23e12 3.0 1
Detail#9Repair#3: 1.73e12 3.0 1.73e12 3.0 1
Detail#9Repair#4: 1.73e12 3.0 5.66e11 3.0 0.1
Detail#9Repair#5: 1.73e12 3.0 1.23e12 3.0 0.1
Detail#9Repair#6: 1.73e12 3.0 1.73e12 3.0 0.1

Detail#10Repair#1: 1.73e12 3.0 5.66e11 3.0 1
Detail#10Repair#2: 1.73e12 3.0 1.23e12 3.0 1
Detail#10Repair#3: 1.73e12 3.0 1.73e12 3.0 1
Detail#10Repair#4: 1.73e12 3.0 5.66e11 3.0 0.6
Detail#10Repair#5: 1.73e12 3.0 1.23e12 3.0 0.6
Detail#10Repair#6: 1.73e12 3.0 1.73e12 3.0 0.6

Detail#11Repair#1: 1.73e12 3.0 5.66e11 3.0 1
Detail#11Repair#2: 1.73e12 3.0 1.23e12 3.0 1
Detail#11Repair#3: 1.73e12 3.0 1.73e12 3.0 1
```

```
Detail#11Repair#4: 1.73e12 3.0 5.66e11 3.0 0.8
Detail#11Repair#5: 1.73e12 3.0 1.23e12 3.0 0.8
Detail#11Repair#6: 1.73e12 3.0 1.73e12 3.0 0.8
```

A.2 REPAIR.DEF

```
NAME        Repair

DESCRIPTION 'Repair Management System for Tankers'

EXETYPE     WINDOWS

STUB        'WINSTUB.EXE'

CODE        PRELOAD MOVEABLE DISCARDABLE
DATA        PRELOAD MOVEABLE MULTIPLE

HEAPSIZE     1024
STACKSIZE    8196

EXPORTS      MainWndProc
             About
```

A.3 REPAIR.H

```
/****************************************************************************

   Module:     Repair.h

   PURPOSE:    To define public structures and associated constants.

****************************************************************************/
/* file menu items */
#define      IDM_NEW      100
#define      IDM_OPEN     101
#define      IDM_SAVE     102
#define      IDM_SAVEAS   103
#define      IDM_EXIT     105

#define  IDC_EDIT  401
#define  IDC_LISTBOX  404
#define  IDC_PATHBOX  406
#define  IDC_PATH  403
#define  Max_FilePath 128
#define  SaveAs_DirLb   8
#define  SaveAs_DirEdit    3
#define  SaveAs_FileEdit  4

/* crack menu items */
#define      IDM_GENERAL  501

/* crack menu items */
#define      IDM_ADDCRACK1  401
#define      IDM_ADDCRACK2  402
#define      IDM_ADDCRACK3  403
#define      IDM_DELETE      404
#define      IDM_EDIT        405

/* analysis menu items */
#define      IDM_Analysis    201
#define       IDPREVIOUS         9
//#define      IDM_FlatbarStiff  202

/* define help items */
#define      IDM_ABOUT                   300
#define      IDM_HELP_INFORMATION        303
#define      IDM_HELP_HOWTO                301

/* define fatigue computing constants */
#define      CyclePerYr              2500000
#define      DamageFactor         1
#define      Uncertainty             1

/* define data size */
#define MaxBitmapfileSize  30000
#define EXE_NAME_MAX_SIZE  128
#define MAX   300
#define MAXG  25
#define MAXL  24
#define MAXD  12
#define MAXLE 10
#define MAXS  25
#define MAXC  25
#define MAXCT 25

/* define or declare structure */
#if fFileStuff
struct  File_Str {
      char Name[14], Path[128];
      OFSTRUCT lpOFSTRUCT;
      };
#if fFileStuffDef
struct File_Str FileStuff;
#else
```

```
struct File_Str extern FileStuff;
#endif
#endif

typedef float near          *PFLOAT;
typedef float near          *NPFLOAT;
typedef float far           *FPFLOAT;
```

A.4 ADDRECRD.C

```
/****************************************************************************

   Module:      AddRecrd.c

   PURPOSE:     Process crack record input dialogue, the Edit and Delete
                 commands on the draw down menu.

   FUNCTIONS:   ShipGeneral()
                 AddRecord()
                 EditRecord()
                 DeleteCrk()

****************************************************************************/
#include <windows.h>
#include <stdlib.h>
#include <string.h>
#include "Repair.h"

char  CrackType[MAXCT];
extern HWND hWnd;
extern HANDLE  hInst;
extern int  fModified, nTotal, nSelect, nAddCrack;
extern int  nX[MAX][3], nY[MAX][3], nClss[MAX];
extern char szName[MAXG], szClass[MAXG], szOwner[MAXG], szClassifi[MAXG],
            szBuilder[MAXG], szDelivery[MAXG], szRoute[MAXG], szOther[MAXG],
            szLocation[MAX][MAXL], szDate[MAX][MAXD], szLength[MAX][MAXLE],
            szStatus[MAX][MAXS], szComment[MAX][MAXC], szCrkType[MAX][MAXCT];
extern void DrawCrack(HWND, int, int, int, BOOL);
extern void DrawIndex(HWND, int, int, BOOL);
extern void EraseIndexSet(HWND, int);
extern void EraseCrackSet(HWND, int);
extern BOOL FAR PASCAL   userPaint(HWND, char*);
extern BOOL FAR PASCAL   myPaint(HWND, char*);
extern BOOL FAR PASCAL   AddRecord(HWND, unsigned, WORD, LONG);
extern BOOL FAR PASCAL   EditRecord(HWND, unsigned, WORD, LONG);
BOOL FAR PASCAL AddGraph(HWND, unsigned, WORD, LONG);
/****************************************************************************

   FUNCTION:    ShipGeneral(HWND, unsigned, WORD, LONG)

   PURPOSE:

   MESSAGES: WM_COMMAND, WM_INITDIALOG

****************************************************************************/
BOOL FAR PASCAL ShipGeneral(hDlg, message, wParam, lParam)
HWND      hDlg;
unsigned message;
WORD      wParam;
LONG      lParam;
{
   switch (message){
     case WM_COMMAND:
        switch (wParam){
           case IDOK:
              GetDlgItemText(hDlg,601, (LPSTR)szName,     MAXG);
              GetDlgItemText(hDlg,602, (LPSTR)szClass,       MAXG);
              GetDlgItemText(hDlg,603, (LPSTR)szOwner,       MAXG);
              GetDlgItemText(hDlg,604, (LPSTR)szClassifi, MAXG);
              GetDlgItemText(hDlg,605, (LPSTR)szBuilder,  MAXG);
              GetDlgItemText(hDlg,606, (LPSTR)szDelivery, MAXG);
              GetDlgItemText(hDlg,607, (LPSTR)szRoute,       MAXG);
              GetDlgItemText(hDlg,608, (LPSTR)szOther,       MAXG);
              EndDialog(hDlg, NULL);
              fModified = TRUE;
              break;

           case IDCANCEL:
              EndDialog(hDlg,NULL);
```

```
                return (TRUE);
            }
          break;

        case WM_INITDIALOG:
          SetDlgItemText(hDlg, 601, (LPSTR) szName);
          SetDlgItemText(hDlg, 602, (LPSTR) szClass);
          SetDlgItemText(hDlg, 603, (LPSTR) szOwner);
          SetDlgItemText(hDlg, 604, (LPSTR) szClassifi);
          SetDlgItemText(hDlg, 605, (LPSTR) szBuilder);
          SetDlgItemText(hDlg, 606, (LPSTR) szDelivery);
          SetDlgItemText(hDlg, 607, (LPSTR) szRoute);
          SetDlgItemText(hDlg, 608, (LPSTR) szOther);
          return (TRUE);

        case WM_CLOSE:
          PostMessage(hDlg, WM_COMMAND, IDCANCEL, 0L);
          break;
        }
    return (FALSE);
}

/****************************************************************************

   FUNCTION:    AddRecord(HWND, unsigned, WORD, LONG)

   PURPOSE:

   MESSAGES: WM_COMMAND, WM_INITDIALOG

****************************************************************************/
BOOL FAR PASCAL AddRecord(hDlg, message, wParam, lParam)
HWND      hDlg;
unsigned message;
WORD      wParam;
LONG      lParam;
{
FARPROC       lpProcAbout;

   switch (message){
      case WM_COMMAND:
         switch (wParam){

            case 607:
             //  ntemp --;
              GetDlgItemText(hDlg,602, (LPSTR)szLocation[nTotal],MAXL);
              GetDlgItemText(hDlg,603, (LPSTR)szDate[nTotal], MAXD);
              GetDlgItemText(hDlg,604, (LPSTR)szLength[nTotal],MAXLE);
              GetDlgItemText(hDlg,605, (LPSTR)szStatus[nTotal], MAXS);
              GetDlgItemText(hDlg,606, (LPSTR)szComment[nTotal],MAXC);

              EndDialog(hDlg, NULL);

              lpProcAbout = MakeProcInstance(AddGraph,hInst);
              DialogBox(hInst, "AddGraphics", hWnd, lpProcAbout);
              FreeProcInstance(lpProcAbout);

              lpProcAbout = MakeProcInstance(AddRecord,hInst);
              DialogBox(hInst, "CrackRecord", hWnd, lpProcAbout);
              FreeProcInstance(lpProcAbout);
              break;

            case IDOK:
              GetDlgItemText(hDlg,602, (LPSTR)szLocation[nTotal],MAXL);
              GetDlgItemText(hDlg,603, (LPSTR)szDate[nTotal], MAXD);
              GetDlgItemText(hDlg,604, (LPSTR)szLength[nTotal],MAXLE);
              GetDlgItemText(hDlg,605, (LPSTR)szStatus[nTotal], MAXS);
              GetDlgItemText(hDlg,606, (LPSTR)szComment[nTotal],MAXC);
              stpcpy(szCrkType[nTotal], CrackType);
              CrackType[0] = 0;
              EndDialog(hDlg, NULL);
              fModified = TRUE;
              break;
```

```
                case IDCANCEL:
                   EndDialog(hDlg,NULL);
                   EraseCrackSet(hWnd, nTotal);
                   CrackType[0] = 0;

                   szCrkType[nTotal][0] = 0;
                   szLocation[nTotal][0] = 0;
                   szDate[nTotal][0]    = 0;
                   szLength[nTotal][0] = 0;
                   szStatus[nTotal][0] = 0;
                   szComment[nTotal][0]= 0;

                   nTotal --;
                   return (TRUE);
                }
             break;

          case WM_INITDIALOG:
             SetDlgItemInt(hDlg, 601, nTotal, TRUE);
             SetDlgItemText(hDlg, 602, (LPSTR) szLocation[nTotal]);
             SetDlgItemText(hDlg, 603, (LPSTR) szDate[nTotal]);
             SetDlgItemText(hDlg, 604, (LPSTR) szLength[nTotal]);
             SetDlgItemText(hDlg, 605, (LPSTR) szStatus[nTotal]);
             SetDlgItemText(hDlg, 606, (LPSTR) szComment[nTotal]);
             return (TRUE);

          case WM_PAINT:
             if(CrackType[0] == 0){
                userPaint(hDlg, CrackType);
                return(TRUE);
                };
             if(userPaint(hDlg, CrackType) == FALSE)
                MessageBox(hDlg,"Cannot find figure.",NULL,MB_OK);
             return (TRUE);

          case WM_CLOSE:
             PostMessage(hDlg, WM_COMMAND, IDCANCEL, 0L);
             break;
          }
       return (FALSE);
}

/****************************************************************************

    FUNCTION:     EditRecord(HWND, unsigned, WORD, LONG)

    PURPOSE:

    MESSAGES: WM_COMMAND, WM_INITDIALOG

****************************************************************************/
BOOL FAR PASCAL EditRecord(hDlg, message, wParam, lParam)
HWND      hDlg;
unsigned  message;
WORD      wParam;
LONG      lParam;
{
FARPROC        lpProcAbout;

   switch (message){
      case WM_COMMAND:
         switch (wParam){

            case 607:
               EndDialog(hDlg, NULL);
               lpProcAbout = MakeProcInstance(AddGraph,hInst);
               DialogBox(hInst, "AddGraphics", hWnd, lpProcAbout);
               FreeProcInstance(lpProcAbout);

               lpProcAbout = MakeProcInstance(EditRecord,hInst);
               DialogBox(hInst, "CrackRecord", hWnd, lpProcAbout);
```

```
                FreeProcInstance(lpProcAbout);
                break;

             case IDOK:
                GetDlgItemText(hDlg,602, (LPSTR)szLocation[nSelect],MAXL);
                GetDlgItemText(hDlg,603, (LPSTR)szDate[nSelect], MAXD);
                GetDlgItemText(hDlg,604, (LPSTR)szLength[nSelect],MAXLE);
                GetDlgItemText(hDlg,605, (LPSTR)szStatus[nSelect], MAXS);
                GetDlgItemText(hDlg,606, (LPSTR)szComment[nSelect],MAXC);
                stpcpy(szCrkType[nSelect], CrackType);
                EndDialog(hDlg, NULL);
                CrackType[0] = 0;
                fModified = TRUE;
                break;

             case IDCANCEL:
                EndDialog(hDlg,NULL);
                CrackType[0] = 0;
                return (TRUE);
             }
          break;

       case WM_INITDIALOG:
          SetDlgItemInt(hDlg, 601, nSelect, TRUE);
          SetDlgItemText(hDlg, 602, (LPSTR) szLocation[nSelect]);
          SetDlgItemText(hDlg, 603, (LPSTR) szDate[nSelect]);
          SetDlgItemText(hDlg, 604, (LPSTR) szLength[nSelect]);
          SetDlgItemText(hDlg, 605, (LPSTR) szStatus[nSelect]);
          SetDlgItemText(hDlg, 606, (LPSTR) szComment[nSelect]);
          return (TRUE);

       case WM_PAINT:
          if(CrackType[0] == 0){ return(TRUE);};
          if(userPaint(hDlg, CrackType) == FALSE)
             MessageBox(hDlg,"Cannot find figure.",NULL,MB_OK);
          return (TRUE);

       case WM_CLOSE:
          PostMessage(hDlg, WM_COMMAND, IDCANCEL, 0L);
          break;
       }
    return (FALSE);
}


/****************************************************************************

   FUNCTION:    DeleteCrk(HWND, unsigned, WORD, LONG)

   PURPOSE:

   MESSAGES: WM_COMMAND, WM_INITDIALOG

****************************************************************************/
BOOL FAR PASCAL DeleteCrk(hDlg, message, wParam, lParam)
HWND      hDlg;
unsigned  message;
WORD      wParam;
LONG      lParam;
{
   inti, tempRec;
   switch (message){
      case WM_COMMAND:
         switch (wParam){
             case IDOK:
                tempRec = nTotal+1;

                nX[tempRec][0] = nX[nSelect][0];
                nY[tempRec][0] = nY[nSelect][0];
                nX[tempRec][1] = nX[nSelect][1];
                nY[tempRec][1] = nY[nSelect][1];
                nX[tempRec][2] = nX[nSelect][2];
                nY[tempRec][2] = nY[nSelect][2];
```


148

```
                nClss[tempRec] = nClss[nSelect];
                for(i=nSelect; i<nTotal; i++){

                   nX[i][0] = nX[i+1][0];
                   nY[i][0] = nY[i+1][0];
                   nX[i][1] = nX[i+1][1];
                   nY[i][1] = nY[i+1][1];
                   nX[i][2] = nX[i+1][2];
                   nY[i][2] = nY[i+1][2];
                   nClss[i] = nClss[i+1];
                   stpcpy(szCrkType[i], szCrkType[i+1]);
                   stpcpy(szLocation[i], szLocation[i+1]);
                   stpcpy(szDate[i], szDate[i+1]);
                   stpcpy(szLength[i], szLength[i+1]);
                   stpcpy(szStatus[i], szStatus[i+1]);
                   stpcpy(szComment[i], szComment[i+1]);
                   };
                if(nTotal == nSelect) {nSelect--;};

                szCrkType[nTotal][0] = 0;
                szLocation[nTotal][0] = 0;
                szDate[nTotal][0]   = 0;
                szLength[nTotal][0] = 0;
                szStatus[nTotal][0] = 0;
                szComment[nTotal][0]= 0;

                nTotal --;
                EndDialog(hDlg, NULL);
                fModified = TRUE;
                EraseIndexSet(hWnd, tempRec);
                EraseCrackSet(hWnd, tempRec);

                break;

             case IDCANCEL:
                EndDialog(hDlg,NULL);
                return (TRUE);
             }
          break;

       case WM_INITDIALOG:
          return (TRUE);

       case WM_CLOSE:
          PostMessage(hDlg, WM_COMMAND, IDCANCEL, 0L);
          break;
       }
   return (FALSE);
}

/****************************************************************************

   FUNCTION:    AddGraph(HWND, unsigned, WORD, LONG)

   PURPOSE:     While users edit crack records, this function pop up a dialog
                box to choose a cracked detail drawing or enter a user bitmap
                name.

   MESSAGES: WM_COMMAND, WM_INITDIALOG

****************************************************************************/
BOOL FAR PASCAL AddGraph(hDlg, message, wParam, lParam)
HWND     hDlg;
unsigned message;
WORD     wParam;
LONG     lParam;
{
   BOOL chk;

   switch (message){
      case WM_COMMAND:
         switch (wParam){
            case 151:
```

```
                case 152:
                case 153:
                case 154:
                case 155:
                case 156:
                case 157:
                case 158:
                case 159:
                case 160:
                case 161:
                case 162:
                case 163:
                case 164:
                   CheckRadioButton(hDlg, 151, 164, wParam);
                   break;

                case IDOK:
                   if(IsDlgButtonChecked(hDlg,151) > 0) stpcpy(CrackType, "crack1.bmp");
                   if(IsDlgButtonChecked(hDlg,152) > 0) stpcpy(CrackType, "crack2.bmp");
                   if(IsDlgButtonChecked(hDlg,153) > 0) stpcpy(CrackType, "crack3.bmp");
                   if(IsDlgButtonChecked(hDlg,154) > 0) stpcpy(CrackType, "crack4.bmp");
                   if(IsDlgButtonChecked(hDlg,155) > 0) stpcpy(CrackType, "crack5.bmp");
                   if(IsDlgButtonChecked(hDlg,156) > 0) stpcpy(CrackType, "crack6.bmp");
                   if(IsDlgButtonChecked(hDlg,157) > 0) stpcpy(CrackType, "crack7.bmp");
                   if(IsDlgButtonChecked(hDlg,158) > 0) stpcpy(CrackType, "crack8.bmp");
                   if(IsDlgButtonChecked(hDlg,159) > 0) stpcpy(CrackType, "crack9.bmp");
                   if(IsDlgButtonChecked(hDlg,160) > 0) stpcpy(CrackType, "crack10.bmp");
                   if(IsDlgButtonChecked(hDlg,161) > 0) stpcpy(CrackType, "crack11.bmp");
                   if(IsDlgButtonChecked(hDlg,162) > 0) stpcpy(CrackType, "crack12.bmp");
                   if(IsDlgButtonChecked(hDlg,163) > 0) stpcpy(CrackType, "crack13.bmp");
                   if(IsDlgButtonChecked(hDlg,164) > 0) {
                      chk = GetDlgItemText(hDlg, 165, (LPSTR) CrackType, 40);
                       if(chk == 0){
                          MessageBox(hDlg, "Invalid Data Input", NULL,
                                          MB_OK | MB_ICONEXCLAMATION);
                       break;
                          };
                       };
                   EndDialog(hDlg,NULL);
                   break;

                case IDCANCEL:
                   EndDialog(hDlg,NULL);
                   return (TRUE);
            }
            break;

        case WM_INITDIALOG:
            CheckRadioButton(hDlg, 151, 163, 163);
            return (TRUE);

        case WM_PAINT:
            myPaint(hDlg, "ADDGRAPHICS");
            return (TRUE);

        case WM_CLOSE:
            PostMessage(hDlg, WM_COMMAND, IDCANCEL, 0L);
            break;
        }
    return (FALSE);
}
```

A.5 ANALYSIS.C

```
/*****************************************************************************

   Module:      Analysis.c

   PURPOSE:      Import and read a user prepared bitmap file.
                 Show data input windows for fatigue life prediction.

   FUNCTIONS:
             userPaint(HWND, char*)
             ReadDIBFile (int)
             MyRead(int, LPSTR, DWORD)
             myPaint()

             RepairMethod()
             Repair456Method()
            UserRprMethod()
            UserInputSN()
             InputYear()
             Analysis()

*****************************************************************************/
#include <windows.h>
#include <stdlib.h>
#include <io.h>
#include 'Repair.h'

# define DIB_HEADER_MARKER   'BM'

int             iDetail;
float        fYr, uA, um, unewA, unewm, ur;
//BOOL             Previous = 0;
extern HANDLE   hInst;               /* Handle to instance data */

extern float CalcFatLife(float, float, float, float, float, float, float);
HANDLE       ReadDIBFile (int hFile);
extern void FAR  ReadRepairData(HWND, int, int, float*, float*, float*, float*,
float*);

/*****************************************************************************

   FUNCTION:    userPaint(HWND, char*)

   PURPOSE:      Import a bitmap file and show it on a window.

*****************************************************************************/
BOOL FAR PASCAL userPaint(hDlg, pmap)
HWND      hDlg;
char      *pmap;
{
   BITMAPFILEHEADER   bmfHeader;
   LPBITMAPINFO       lpBitmapInfo;
   LPSTR                 lpBits;
   static int            nColorData;
   int                hFile, hDiskbuf;
   LPBITMAPFILEHEADER lpBitmapFileHeader;
   PAINTSTRUCT           Paint;
   OFSTRUCT              of;
   LPSTR                 Diskbuf;
   HANDLE             hDIB;
   unsigned int       numread;
   LPSTR              lpDIBHdr, lpDIBBits;

   BeginPaint(hDlg, &Paint);
   if ((hFile = OpenFile (pmap, &of, OF_READ)) == -1)
      {
      EndPaint(hDlg, &Paint);
      return FALSE;
      }
   else
```

```
    {
    if (!(hDIB = ReadDIBFile (hFile)))
        {
        _lclose (hFile);
        MessageBox(hDlg,"Not enough memory to read file",NULL,
                                 MB_OK|MB_ICONEXCLAMATION);
    return FALSE;
    }

  lpDIBHdr  = GlobalLock (hDIB);
    lpBitmapInfo = (LPBITMAPINFO) &lpDIBHdr[14];
    if(lpBitmapInfo->bmiHeader.biClrUsed != 0)
       nColorData = lpBitmapInfo->bmiHeader.biClrUsed;
    else   {
           switch(lpBitmapInfo->bmiHeader.biBitCount)
              {
              case 1:   nColorData = 2;
                        break;
              case 4:   nColorData = 16;
                        break;
              case 8:   nColorData = 256;
                        break;
              case 24:  nColorData = 0;
                        break;
              }
       }
    lpBits = (LPSTR) lpBitmapInfo;
    lpBits+=(WORD)lpBitmapInfo->bmiHeader.biSize+(WORD)(nColorData*sizeof(RGBQUAD));

    SetDIBitsToDevice (Paint.hdc,                    // hDC
                        (WORD) 0,                  // DestX
                        (WORD) 0,                    // DestY
         (WORD) lpBitmapInfo->bmiHeader.biWidth,
         (WORD) lpBitmapInfo->bmiHeader.biHeight,
         (WORD) 0, (WORD) 0, (WORD) 0,
         (WORD) lpBitmapInfo->bmiHeader.biHeight,
                        lpBits,                      // lpBits
         (LPBITMAPINFO) lpBitmapInfo,
                        DIB_RGB_COLORS);             // wUsage

     SetDIBitsToDevice(Paint.hdc, 0, 0,
     (WORD) lpBitmapInfo->bmiHeader.biWidth,
     (WORD) lpBitmapInfo->bmiHeader.biHeight,
     0, 0, 0,
     (WORD) lpBitmapInfo->bmiHeader.biHeight,
     lpBits, lpBitmapInfo, DIB_RGB_COLORS);

     _lclose(hFile);
     GlobalUnlock(hDIB);
     GlobalDiscard(hDIB);
     EndPaint(hDlg, &Paint);
     return TRUE;
     }
}
/*************************************************************************

  Function:  ReadDIBFile (int)
   Purpose:  Reads in the specified DIB file into a global chunk of
             memory.
   Returns:  A handle to a dib (hDIB) if successful.
             NULL if an error occurs.
  Comments:  BITMAPFILEHEADER is stripped off of the DIB.  Everything
             from the end of the BITMAPFILEHEADER structure on is
             returned in the global memory handle.

*************************************************************************/
HANDLE ReadDIBFile (int hFile)
{
   BITMAPFILEHEADER    bmfHeader;
   DWORD               dwBitsSize;
   HANDLE              hDIB;
   LPSTR               pDIB;
```


152

```
   // get length of DIB in bytes for use when reading
   dwBitsSize = filelength (hFile);

   // Allocate memory for DIB
   hDIB = GlobalAlloc (GMEM_MOVEABLE | GMEM_ZEROINIT, dwBitsSize -
sizeof(BITMAPFILEHEADER));
   pDIB = GlobalLock (hDIB);

   // Go read the bits.
   if (!MyRead (hFile, pDIB, dwBitsSize - sizeof(BITMAPFILEHEADER)))
      {
      GlobalUnlock (hDIB);
      GlobalFree   (hDIB);
      return NULL;
      }

   GlobalUnlock (hDIB);
   return hDIB;
}

/***************************************************************************

  Function:  MyRead (int, LPSTR, DWORD)
   Purpose:  Routine to read files greater than 64K in size.
   Returns:  TRUE if successful.
             FALSE if an error occurs.

***************************************************************************/
# define  BYTES_PER_READ 32767

BOOL MyRead (int hFile, LPSTR lpBuffer, DWORD dwSize)
{
   char huge *lpInBuf = (char huge *) lpBuffer;
   int        nBytes;


   while (dwSize)
      {
      nBytes = (int) (dwSize > (DWORD) BYTES_PER_READ ? BYTES_PER_READ :
                                                        LOWORD (dwSize));

      if (_lread (hFile, (LPSTR) lpInBuf, nBytes) != (WORD) nBytes)
         return FALSE;

      dwSize  -= nBytes;
      lpInBuf += nBytes;
      }

   return TRUE;
}
/****************************************************************************

   FUNCTION:    myPaint(HWND, char*)

   PURPOSE:     Show a resource bitmap on a dialog box.

   MESSAGES: WM_COMMAND, WM_INITDIALOG

****************************************************************************/
BOOL FAR PASCAL myPaint(hDlg, pmap)
HWND      hDlg;
char      *pmap;
{
   PAINTSTRUCT Paint;
   HBITMAP  hButton;
   HDC hMemoryDC;
   HANDLE hDefault;

   FARPROC lpProcAbout;
   int i;
```

```
        BeginPaint(hDlg, &Paint);
        hMemoryDC = CreateCompatibleDC(Paint.hdc);
        hButton = LoadBitmap(hInst, pmap);
        hDefault = SelectObject(hMemoryDC, hButton);
        BitBlt(Paint.hdc, Paint.rcPaint.left, Paint.rcPaint.top,
            (Paint.rcPaint.right - Paint.rcPaint.left),
            (Paint.rcPaint.bottom - Paint.rcPaint.top),
            hMemoryDC, Paint.rcPaint.left, Paint.rcPaint.top, SRCCOPY);
        SelectObject(hMemoryDC, hDefault);
        DeleteObject(hButton);
        DeleteDC(hMemoryDC);
        EndPaint(hDlg, &Paint);
        return(TRUE);
}
/****************************************************************************

   FUNCTION:    RepairMethod(HWND, unsigned, WORD, LONG)

   PURPOSE:     Process a dialog box that let user select a repair method.

   MESSAGES: WM_COMMAND, WM_INITDIALOG

****************************************************************************/
BOOL FAR PASCAL RepairMethod(hDlg, message, wParam, lParam)
HWND      hDlg;
unsigned message;
WORD      wParam;
LONG      lParam;
{
   FARPROC lpProcAbout;
   int i;
   float A, m, newA, newm, r;
   char resultstr[20];

   switch (message){
      case WM_COMMAND:
         switch (wParam){
            case IDOK:
               EndDialog(hDlg,NULL);
               break;

            case IDCANCEL:
               EndDialog(hDlg,NULL);
               return (TRUE);
         }
         break;

      case WM_INITDIALOG:
         for (i=1; i <= 6; i++)
            {
            if(iDetail != 12)
            {
               ReadRepairData(hDlg, iDetail, i, &A, &m, &newA, &newm, &r);
               };
            gcvt(CalcFatLife(fYr, 0.9, A, m, newA, newm, r), 5, resultstr);
            SetDlgItemText(hDlg, 600+i, (LPSTR) resultstr);
            };
         break;

      case WM_PAINT:
         switch (iDetail){
                  case 1:
                     myPaint(hDlg, "LONGIREPAIR1");
                     break;
                  case 2:
                     myPaint(hDlg, "LONGIREPAIR2");
                     break;
                  case 3:
                     myPaint(hDlg, "LONGIREPAIR3");
                     break;
                  case 7:
                     myPaint(hDlg, "FLATBARREPAIR7");
```

```
                        break;
                    case 8:
                        myPaint(hDlg, "FLATBARREPAIR8");
                        break;
                    case 9:
                        myPaint(hDlg, "FLATBARREPAIR9");
                        break;
                    case 10:
                        myPaint(hDlg, "BEAMREPAIR10");
                        break;
                    case 11:
                        myPaint(hDlg, "BEAMREPAIR11");
                        break;
                    };
            return (TRUE);

        case WM_CLOSE:
            PostMessage(hDlg, WM_COMMAND, IDPREVIOUS, 0L);
            break;
        }
    return (FALSE);
}

/****************************************************************************

    FUNCTION:    Repair456Method(HWND, unsigned, WORD, LONG)

    PURPOSE:     Process a dialog box that let user select a repair method.

    MESSAGES: WM_COMMAND, WM_INITDIALOG

****************************************************************************/
BOOL FAR PASCAL Repair456Method(hDlg, message, wParam, lParam)
HWND      hDlg;
unsigned message;
WORD      wParam;
LONG      lParam;
{
    FARPROC lpProcAbout;
    int i;
    float A, m, newA, newm, r;
    char resultstr[20];

    switch (message){
        case WM_COMMAND:
            switch (wParam){
                case IDOK:
                    EndDialog(hDlg,NULL);
                    break;

                case IDCANCEL:
                    EndDialog(hDlg,NULL);
                    return (TRUE);
            }
            break;

        case WM_INITDIALOG:
            for (i=1; i <= 3; i++)
                {
                if(iDetail != 12)
                {
                    ReadRepairData(hDlg, iDetail, i, &A, &m, &newA, &newm, &r);
                   };
                gcvt(CalcFatLife(fYr, 0.9, A, m, newA, newm, r), 5, resultstr);
                SetDlgItemText(hDlg, 600+i, (LPSTR) resultstr);
                };
            break;

        case WM_PAINT:
            switch (iDetail){
                    case 4:
                        myPaint(hDlg, "LONGIREPAIR4");
                        break;
```

```
                case 5:
                    myPaint(hDlg, "LONGIREPAIR5");
                    break;
                case 6:
                    myPaint(hDlg, "LONGIREPAIR6");
                    break;
                };
            return (TRUE);

        case WM_CLOSE:
            PostMessage(hDlg, WM_COMMAND, IDPREVIOUS, 0L);
            break;
        }
    return (FALSE);
}

/****************************************************************************

    FUNCTION:   UserRprMethod(HWND, unsigned, WORD, LONG)

    PURPOSE:    Process a dialog box that let user select a repair method.

    MESSAGES: WM_COMMAND, WM_INITDIALOG

****************************************************************************/
BOOL FAR PASCAL UserRprMethod(hDlg, message, wParam, lParam)
HWND     hDlg;
unsigned message;
WORD     wParam;
LONG     lParam;
{
    FARPROC lpProcAbout;
    int i;
    char resultstr[20];

    switch (message){
        case WM_COMMAND:
            switch (wParam){
                case IDOK:
                    EndDialog(hDlg,NULL);
                    break;

                case IDCANCEL:
                    EndDialog(hDlg,NULL);
                    return (TRUE);
            }
            break;

        case WM_INITDIALOG:
            gcvt(CalcFatLife(fYr, 0.9, uA,um,unewA,unewm,ur), 5, resultstr);
            SetDlgItemText(hDlg, 601, (LPSTR) resultstr);
            break;

        case WM_CLOSE:
            PostMessage(hDlg, WM_COMMAND, IDPREVIOUS, 0L);
            break;
        }
    return (FALSE);
}


/****************************************************************************

    FUNCTION:   UserInputSN(HWND, unsigned, WORD, LONG)

    PURPOSE:    Process dialog box that let user select a cracking zone.

    MESSAGES: WM_COMMAND, WM_INITDIALOG

****************************************************************************/
BOOL FAR PASCAL UserInputSN(hDlg, message, wParam, lParam)
HWND     hDlg;
unsigned message;
```

```
WORD      wParam;
LONG      lParam;
{
   FARPROC lpProcAbout;
   char sztemp[5][15];
   int i;
   BOOL chk;

   switch (message){
      case WM_COMMAND:
         switch (wParam){
            case 113:
            case 114:
            case 115:
            case 116:
            case 117:
               break;

            case IDOK:
               for(i=0; i<=4; i++)
               {
                  chk = GetDlgItemText(hDlg, 113+i, (LPSTR) sztemp[i], 15);
                  if(chk == 0)
                     {
                     MessageBox(hDlg, 'Invalid Data Input', NULL,
                                    MB_OK | MB_ICONEXCLAMATION);

                     break;
                     };
                  };
               uA = atof(sztemp[0]);
               um = atof(sztemp[1]);
               unewA = atof(sztemp[2]);
               unewm = atof(sztemp[3]);
               ur = atof(sztemp[4]);

               lpProcAbout = MakeProcInstance(UserRprMethod, hInst);
               DialogBox(hInst, 'UserRprMethodDlg', hDlg, lpProcAbout);
               FreeProcInstance(lpProcAbout);
               EndDialog(hDlg, NULL);
               break;

            case IDCANCEL:
               EndDialog(hDlg,NULL);
               return (TRUE);
            }
         break;

      case WM_INITDIALOG:
         return (TRUE);

      case WM_CLOSE:
         PostMessage(hDlg, WM_COMMAND, IDCANCEL, 0L);
         break;
      }
   return (FALSE);
}
/****************************************************************************

   FUNCTION:   InputYear(HWND, unsigned, WORD, LONG)

   PURPOSE:    Process dialog box that let user select a cracking zone.

   MESSAGES: WM_COMMAND, WM_INITDIALOG

****************************************************************************/
BOOL FAR PASCAL InputYear(hDlg, message, wParam, lParam)
HWND      hDlg;
unsigned  message;
WORD      wParam;
LONG      lParam;
{
   FARPROC lpProcAbout;
```

```
    char szFailYr[6];
    BOOL chk;

    switch (message){
       case WM_COMMAND:
          switch (wParam){
             case 113:
                break;

             case IDOK:
                chk = GetDlgItemText(hDlg, 113, (LPSTR) szFailYr, 6);
                if(chk == 0){
                   MessageBox(hDlg, "Invalid Data Input", NULL,
                                         MB_OK | MB_ICONEXCLAMATION);
                  break;
                  }
                fYr = atof(szFailYr);

                if(iDetail == 12)  /* User Input SN information */
                   {
                   lpProcAbout = MakeProcInstance(UserInputSN,hInst);
                   DialogBox(hInst, "UserInputSNDlg", hDlg, lpProcAbout);
                   FreeProcInstance(lpProcAbout);
                   EndDialog(hDlg,NULL);
                   }
                else if((iDetail >= 4) && (iDetail <= 6))
                   {
                   lpProcAbout = MakeProcInstance(Repair456Method,hInst);
                   DialogBox(hInst, "Repair456MethodDlg", hDlg, lpProcAbout);
                   FreeProcInstance(lpProcAbout);
                   EndDialog(hDlg,NULL);
                  }
                else
                {
                   lpProcAbout = MakeProcInstance(RepairMethod,hInst);
                   DialogBox(hInst, "RepairMethodDlg", hDlg, lpProcAbout);
                   FreeProcInstance(lpProcAbout);
                   EndDialog(hDlg,NULL);
                };

                break;

             case IDCANCEL:
                EndDialog(hDlg,NULL);
                return (TRUE);
             }
          break;

      case WM_INITDIALOG:
         return (TRUE);

      case WM_CLOSE:
         PostMessage(hDlg, WM_COMMAND, IDCANCEL, 0L);
         break;
      }
   return (FALSE);
}

/****************************************************************************

   FUNCTION:     Analysis(HWND, unsigned, WORD, LONG)

   PURPOSE:      Process dialog box that let user select
                 a Longitudianal cutout type.

   MESSAGES: WM_COMMAND, WM_INITDIALOG

****************************************************************************/
BOOL FAR PASCAL Analysis(hDlg, message, wParam, lParam)
HWND      hDlg;
unsigned  message;
WORD      wParam;
LONG      lParam;
```


158

```
{
   FARPROC lpProcAbout;
   int i;

   switch (message){
      case WM_COMMAND:
         switch (wParam){
            case 121: /* Twelve radio buttons one is on and others are off.*/
            case 122:
            case 123:
            case 124:
            case 125:
            case 126:
            case 127:
            case 128:
            case 129:
            case 130:
            case 131:
            case 132:
               CheckRadioButton(hDlg, 121, 132, wParam);
               break;

            case IDOK:
               for (i=121; i <= 132; i++)
                  {
                  if(IsDlgButtonChecked(hDlg, i) > 0) iDetail = i - 120;
                  };

               lpProcAbout = MakeProcInstance(InputYear,hInst);
               DialogBox(hInst, "InputYearDlg", hDlg, lpProcAbout);
               FreeProcInstance(lpProcAbout);
               EndDialog(hDlg,NULL);

               break;

            case IDCANCEL:
               EndDialog(hDlg,NULL);
               return (TRUE);
            }
         break;

      case WM_INITDIALOG:
         CheckRadioButton(hDlg, 121, 132, 121);
         return (TRUE);

      case WM_PAINT:
         myPaint(hDlg, "DETAILS12");
         return (TRUE);

      case WM_CLOSE:
         PostMessage(hDlg, WM_COMMAND, IDPREVIOUS, 0L);
         break;
      }
   return (FALSE);
}
```

A.6 CALCUFAT.C

```
/*****************************************************************************

   Module:     Calcufat.c

   PURPOSE:    Calculate fatigue life.

   FUNCTIONS:
   LOCAL:    gammln()
             StressS0()
             FatigueLife()
             CalcFatLife()

*****************************************************************************/
#include <windows.h>
#include <math.h>
#include "Repair.h"

/*****************************************************************************

   FUNCTION: float gammln(float)

   PURPOSE:      Return the value ln[gamma(xx)] for xx>0.  Full accuracy is
                 obtained for xx>1.

   Source:       This function is adapted from "Numerical Recipes in C: The
                 Art of Scientific Computing", William H. Press,
                 Brian P. Flannery, Saul A. Teukolsky, William T. Vetterling.

*****************************************************************************/
float gammln(xx)
float xx;
{
   double x, tmp, ser;
   static double cof[6]={76.18009173, -86.50532033, 24.01409822, -1.231739516,
                         0.120858003e-2, -0.536382e-5};
   int j;

   x=xx-1.0;
   tmp=x+5.5;
   tmp -= (x+0.5)*log(tmp);
   ser=1.0;
   for(j=0;j<=5;j++) {
      x += 1.0;
      ser += cof[j]/x;
      }
   return -tmp+log(2.50662827465*ser);
}


/*****************************************************************************

   FUNCTION:    float StressS0(float, float, float, float)

   PURPOSE:     Calculate extreme stress 'S0'

*****************************************************************************/
float StressS0(float FailYear, float Weibull,float LifeInter,float SNInvSlop)
{
   float Stress;

   Stress = pow( (log(CyclePerYr*FailYear)) , (1/Weibull) ) /Uncertainty
            * pow( (DamageFactor*LifeInter/CyclePerYr/FailYear/
            exp(gammln(SNInvSlop/Weibull+1))) , (1/SNInvSlop) );

   return Stress;
}


/*****************************************************************************
```


160

```
   FUNCTION:     float FatigueLife(float, float, float, float, float)

   PURPOSE:      Calculate expected fatigue life

******************************************************************************/
float FatigueLife(float StressS0, float FailYear, float Weibull,
                  float NewLifeInter, float NewSNInvSlop)
{
   float Life;

   Life = DamageFactor
            * NewLifeInter
            * pow( (log(CyclePerYr*FailYear)), (NewSNInvSlop/Weibull) )
            / CyclePerYr
            / pow( (Uncertainty*StressS0), NewSNInvSlop)
            / exp(gammln(NewSNInvSlop/Weibull+1));

   return Life;
}

/******************************************************************************

   FUNCTION:     float CalcFatLife()

   PURPOSE:      Process S-N data and detail types etc. in order to compute
            fatigue life.

   MESSAGES:

******************************************************************************/
float CalcFatLife(float FYr, float Weib, float A, float m, float newA,
                  float newm, float r)
{
   float tempS0, tempL, tempL2;

   tempS0 = StressS0(FYr, Weib, A, m);
   tempS0 = tempS0 * r;

   /* Begin to Iteration */
   tempL = FatigueLife(tempS0, FYr, Weib, newA, newm);

   if (((A==newA) && (m==newm)) && (r==1))
         {return FYr;}
   else  {  do {
               tempL2 = FatigueLife(tempS0, tempL, Weib, newA, newm);
               tempL = FatigueLife(tempS0, tempL2, Weib, newA, newm);
               } while((tempL-tempL2)<0.01 && (tempL-tempL2)>-0.01 );

         return tempL;
         }
}
```

A.7 FILECMDS.C

```
/****************************************************************************

   Module:      FileCMDs.C

   PURPOSE:     Performs File Commands for the top Menu Bar

   FUNCTIONS:
   EXPORTED:
      OpenSource() - Processes Dialog messages.
      SaveAs() - Processes Dialog messages.
      SaveFile() - Processes Dialog messages
      NewFile() - Processes Dialog messages
      SelectLayout()

****************************************************************************/
#define fFileStuff    TRUE
#define fFileStuffDef TRUE

#include <string.h>
#include <windows.h>
#include <io.h>
#include <stdio.h>
#include "repair.h"

BOOL       userBitmap;

int              fModified;
char             ShipType[20], OldType[20];
extern int       nTotal;
extern HWND  hWnd;
extern HANDLE    hInst;          /* Handle to instance data*/
extern char      szName[MAXG], szClass[MAXG], szOwner[MAXG], szClassifi[MAXG],
            szBuilder[MAXG], szDelivery[MAXG], szRoute[MAXG], szOther[MAXG];

char FileName[Max_FilePath];
char PathName[Max_FilePath];
char OpenPath[Max_FilePath];        /* exported for Main window */
char DefPath[Max_FilePath];         /* Init in main from .exe file path. */
char DefSpec[14] = "*.RMS";
char DefExt[] = ".RMS";
char str[Max_FilePath + 14];
char UpTree[] = "..\\";         /* this is a special case where we go up the
                                   Search tree one level*/
char const DirectoryDel[] = "\\";
char const FileSignature[] = "REPAIR MANAGEMENT SYSTEM DATA FILE.";
                                /* open file error messages */
char const FormatError[] = "Format Unknown!";
char const NoFileErr[] = "No File Specified!";
char const UnableOpenEr[] = "Unable to Open File!";
char const FileExists[] = "File Exists. Overwrite?";

extern void FAR UpdateListBox (HWND, LPSTR, LPSTR, LPSTR);
extern void FAR SeparateFile (LPSTR, LPSTR, LPSTR);
extern void FAR ChangeDefExt (LPSTR, LPSTR);
extern void FAR UpTreePath (LPSTR);
extern int  FAR VerifySig (int, LPSTR);
extern void FAR WriteText(HWND);
extern void FAR ReadNativeFormat(HWND);
extern BOOL FAR PASCAL myPaint(HWND, char*);

/****************************************************************************

   FUNCTION:   OpenSource(HWND, unsigned, WORD, LONG)

   PURPOSE:    Processes Dialog messages.

   MESSAGES: WM_COMMAND, WM_INITDIALOG

****************************************************************************/
```

```
LONG FAR PASCAL OpenSource(hDlg, message, wParam, lParam)
HWND        hDlg;
unsigned    message;
WORD        wParam;
LONG        lParam;
{

   WORD index;
   PSTR pTptr;
   HANDLE hFile;
   OFSTRUCT lpOFSTRUCT;

   switch (message){
      case WM_COMMAND:
          switch (wParam){
            case IDC_LISTBOX:
               switch (HIWORD(lParam)) {
                  case LBN_SELCHANGE:
                     if (!DlgDirSelect(hDlg, FileName, IDC_LISTBOX)) {
                        SetDlgItemText(hDlg,IDC_EDIT, FileName);
                        SendDlgItemMessage(hDlg, IDC_EDIT,EM_SETSEL,
                           NULL, MAKELONG(0, 0x7fff));
                     }
                     break;
                  case LBN_DBLCLK:
                     goto openfile;
               }
               return (TRUE);
/* */
            case IDC_PATHBOX:
               switch (HIWORD(lParam)) {
                  case LBN_SELCHANGE:
                     SeparateFile((LPSTR) PathName,
                          (LPSTR) FileName, (LPSTR) str);
                     DlgDirSelect(hDlg, str, IDC_PATHBOX);
                     if (_fstrcmp(UpTree, str) == 0){
                        UpTreePath(PathName);
                        _fstrcpy(str, PathName);
                     } else if (str[_fstrlen(str)-1] == ':') {
                        _fstrcat(str, DirectoryDel);
                     } else if (str[_fstrlen(str)-1] == '\\'){
                        _fstrcat(PathName, str);
                        _fstrcpy(str, PathName);
                     } else {
                     /* This is for a strange case where a user double
                        clicks in the path box causing the entire
                        path string to be returned */
                        _fstrcpy(str, PathName);
                     }
                     SetDlgItemText(hDlg, IDC_PATH, str);
                     _fstrcat(str, FileName);
                     _fstrcpy(FileName, str);

                     /* Update file list from new directory, Revised */
                     DlgDirList(hDlg, FileName, IDC_LISTBOX,
                         IDC_PATH, 0);
                     if (_fstrchr(FileName, '*') || _fstrchr(FileName, '?'))
                     {
                        SeparateFile((LPSTR) DefPath, (LPSTR) PathName,
                           (LPSTR) str);
                        _fstrcpy(DefSpec, FileName);
                        ChangeDefExt(DefExt, DefSpec);
                        UpdateListBox(hDlg, DefPath, FileName, str);
                        return(TRUE);
                     };
                     /* Update Path list box */
                     _fstrcpy(FileName, PathName);
                     _fstrcat(str, DefSpec);
                     DlgDirList(hDlg, FileName, IDC_PATHBOX,IDC_PATH, 0xC010);

                     SendDlgItemMessage(hDlg, IDC_EDIT,EM_SETSEL,
                        NULL, MAKELONG(0, 0x7fff));
                     break;
```

```
                    case LBN_DBLCLK:
                        break;
                }
                return (TRUE);

            case IDOK:
openfile:
                GetDlgItemText(hDlg, IDC_EDIT, FileName, 128);
                if (_fstrchr(FileName, '*') ||
                    _fstrchr(FileName, '?')) {
                    SeparateFile((LPSTR) DefPath, (LPSTR) PathName,
                        (LPSTR) str);
                    _fstrcpy(DefSpec, FileName);

                    ChangeDefExt(DefExt, DefSpec);
                    UpdateListBox(hDlg, DefPath, FileName, str);
                    return(TRUE);
                }
                if (!FileName[0]) {
                    MessageBox(hDlg, (LPSTR) NoFileErr,
                        (LPSTR) NULL, MB_OK | MB_ICONQUESTION);
                    return (TRUE);
                }
                /* at this point we have a real file name! */
                SeparateFile((LPSTR) DefPath, (LPSTR) PathName,
                    (LPSTR) str);
                _fstrcpy(str, DefPath);
                _fstrcat(str, FileName);
                hFile = OpenFile(str, &lpOFSTRUCT, OF_READ);
                if (hFile == 0xFFFF) {
                    MessageBox(hDlg, (LPSTR) UnableOpenEr,
                        (LPSTR) NULL, MB_OK | MB_ICONQUESTION);
                    return (TRUE);
                }

                if(!VerifySig(hFile, (LPSTR) FileSignature)) {
                    MessageBox(hDlg, (LPSTR) FormatError,
                        (LPSTR) NULL, MB_OK | MB_ICONSTOP | MB_TASKMODAL);
                    close (hFile);
                    /* make sure handle is invalid to prevent reopen.*/
                    hFile = 0;
                    return (TRUE);
                }
                close (hFile); /* leave file closed and reopen when needed*/
                _fstrcpy(FileStuff.Name, FileName);
                _fstrcpy(FileStuff.Path, DefPath);
                memcpy(&FileStuff.lpOFSTRUCT, &lpOFSTRUCT, sizeof(OFSTRUCT));
                ReadNativeFormat(hDlg);
                fModified = FALSE;
                InvalidateRect(hWnd, NULL, 1);
                PostMessage(hWnd, WM_PAINT, NULL, NULL);

            case IDCANCEL:
                EndDialog(hDlg,NULL);
                return (TRUE);

            default:
                return (TRUE);
        }

    case WM_INITDIALOG:
        UpdateListBox(hDlg, DefPath, DefSpec, str);
        SendDlgItemMessage(hDlg, IDC_EDIT, EM_SETSEL, NULL,
            MAKELONG(0, 0x7fff));
        SetFocus(GetDlgItem(hDlg, IDC_EDIT));
        return (FALSE);
    }
    return (FALSE);
}

/****************************************************************************
```


164

```
FUNCTION:    SaveAs(HWND, unsigned, WORD, LONG)

PURPOSE:     Processes Dialog messages.

MESSAGES:

****************************************************************************/
LONG FAR PASCAL SaveAs(hDlg, message, wParam, lParam)
HWND        hDlg;
unsigned    message;
WORD        wParam;
LONG        lParam;
{
    HANDLE hFile;
    OFSTRUCT lpOFSTRUCT;

    switch (message){
        case WM_COMMAND:
            switch (wParam){
              case SaveAs_DirLb:
                 switch (HIWORD(lParam)) {
                    case LBN_SELCHANGE:
                       DlgDirSelect(hDlg, str, SaveAs_DirLb);
                       if (_fstrcmp(UpTree, str) == 0){
                          UpTreePath(PathName);
                          _fstrcpy(str, PathName);
                       } else if (str[_fstrlen(str)-1] == ':') {
                          _fstrcat(str, DirectoryDel);
                       } else if (str[_fstrlen(str)-1] == '\\'){
                          _fstrcat(PathName, str);
                          _fstrcpy(str, PathName);
                       } else {
                       /* This is for a strange case where a user double
                          clicks in the path box causing the entire
                          path string to be returned */
                          _fstrcpy(str, PathName);
                       }
                       _fstrcpy(PathName, str);
                       _fstrcat(str, DefSpec);

                       /* Update Path list box */
                       DlgDirList(hDlg, str, SaveAs_DirLb,
                             SaveAs_DirEdit, 0xC010);

                       break;
                     case LBN_DBLCLK:
                       break;
                  }
                  return (TRUE);

               case IDOK:
/* openfile:   */
                  GetDlgItemText(hDlg, SaveAs_FileEdit, FileName, 128);
                  GetDlgItemText(hDlg, SaveAs_DirEdit, PathName, 128);
                  _fstrcat(PathName, DirectoryDel);
                  if (_fstrchr(FileName, '*') ||
                     _fstrchr(FileName, '?')) {
                     MessageBox(hDlg, "Wildcard inappropriate.",
                        NULL, MB_OK | MB_ICONQUESTION);
                     return(TRUE);
                  }
                  if (!FileName[0]) {
                     MessageBox(hDlg, (LPSTR) NoFileErr,
                        (LPSTR) NULL, MB_OK | MB_ICONQUESTION);
                     return (TRUE);
                  }
                  if (_fstrchr(FileName, '.')) {
                     MessageBox(hDlg, "File Name should not have an Extention.",
                        NULL, MB_OK | MB_ICONEXCLAMATION);
                     *_fstrchr(FileName, '.') = 0;/* remove extention */
                     SetDlgItemText(hDlg, SaveAs_FileEdit, FileName);
                     return(TRUE);
                  }
```

```
            if (PathName[_fstrlen(PathName)-1] != '\\')
               _fstrcat(PathName, DirectoryDel);
            _fstrcpy(DefPath, PathName);
            _fstrcpy(str, PathName);
            _fstrcat(FileName, ".rms");
            _fstrcat(str, FileName);
            hFile = OpenFile(str, &lpOFSTRUCT, OF_EXIST);
            if (hFile != 0xFFFF) {
               if (MessageBox(hDlg, (LPSTR) FileExists, (LPSTR) FileName,
                  MB_OKCANCEL | MB_ICONQUESTION) == IDCANCEL)
                  return (TRUE);
            }
            hFile = OpenFile(str, &lpOFSTRUCT, OF_CREATE);
            if (hFile == 0xFFFF) {
               MessageBox(hDlg, (LPSTR) UnableOpenEr,
                  (LPSTR) NULL, MB_OK | MB_ICONQUESTION);
               return (TRUE);
            }
            close (hFile);     /* leave file closed and reopen when needed*/
            _fstrcpy(FileStuff.Name, FileName);
            _fstrcpy(FileStuff.Path, PathName);
            memcpy(&FileStuff.lpOFSTRUCT, &lpOFSTRUCT, sizeof(OFSTRUCT));

            WriteText(hDlg);

         case IDCANCEL:
            EndDialog(hDlg,NULL),
            return (TRUE);

         default:
            return (TRUE);
      }

   case WM_INITDIALOG:
      _fstrcpy(str, DefPath);
      _fstrcpy(FileName, FileStuff.Name);
      *_fstrchr(FileName, '.') = 0;             /* remove extention */
      _fstrcat(str, DefSpec);
      _fstrcpy(PathName, DefPath);
      /* Update Path list box */
      DlgDirList(hDlg, str, SaveAs_DirLb, SaveAs_DirEdit, 0xC010);
      SetDlgItemText(hDlg, SaveAs_FileEdit, FileName);
      SetFocus(GetDlgItem(hDlg, SaveAs_FileEdit));
      return (FALSE);
   }
return (NULL);
}
/****************************************************************************

   FUNCTION:   SaveFile(HWND)

   PURPOSE:    Processes Dialog messages.

****************************************************************************/
VOID FAR SaveFile(hWnd)
HWND      hWnd;
{
   WriteText(hWnd);          /* this will write the current file out */
}
/****************************************************************************

   FUNCTION:   NewFile(HWND, unsigned, WORD, LONG)

   PURPOSE:    Processes Dialog messages.

   MESSAGES:

****************************************************************************/

BOOL FAR PASCAL NewFile(hDlg, message, wParam, lParam)
HWND       hDlg;
```

```
unsigned    message;
WORD        wParam;
LONG        lParam;
{
   HANDLE hFile;
   OFSTRUCT lpOFSTRUCT;

   switch (message){
      case WM_COMMAND:
           switch (wParam){
             case SaveAs_DirLb:
                switch (HIWORD(lParam)) {
                   case LBN_SELCHANGE:
                      DlgDirSelect(hDlg, str, SaveAs_DirLb);
                      if (_fstrcmp(UpTree, str) == 0){
                         UpTreePath(PathName);
                         _fstrcpy(str, PathName);
                      } else if (str[_fstrlen(str)-1] == ':') {
                         _fstrcat(str, DirectoryDel);
                      } else if (str[_fstrlen(str)-1] == '\\'){
                         _fstrcat(PathName, str);
                         _fstrcpy(str, PathName);
                      }
                      _fstrcpy(PathName, str);
                      _fstrcat(str, DefSpec);

                      /* Update Path list box */
                      DlgDirList(hDlg, str, SaveAs_DirLb,
                            SaveAs_DirEdit, 0xC010);
                      break;
                    case LBN_DBLCLK:
                      break;
                }
                return (TRUE);

              case IDOK:
/* openfile:  */
                GetDlgItemText(hDlg, SaveAs_FileEdit, FileName, 128);
                GetDlgItemText(hDlg, SaveAs_DirEdit, PathName, 128);
                if (!FileName[0]) {
                   MessageBox(hDlg, (LPSTR) NoFileErr,
                      (LPSTR) NULL, MB_OK | MB_ICONQUESTION);
                   return (TRUE);
                 }
                if (_fstrchr(FileName, '*') ||
                   _fstrchr(FileName, '?')) {
                   MessageBox(hDlg, "Wildcard inappropriate.",
                      NULL, MB_OK | MB_ICONQUESTION);
                   return(TRUE);
                }
                if (_fstrchr(FileName, '.') == 0)/* no extention */
                   _fstrcat(FileName, DefExt); /* add default */
                if (PathName[_fstrlen(PathName)-1] != '\\')
                   _fstrcat(PathName, DirectoryDel);
                _fstrcpy(DefPath, PathName);
                _fstrcpy(str, PathName);
                _fstrcat(str, FileName);
                hFile = OpenFile(str, &lpOFSTRUCT, OF_EXIST);
                if (hFile != 0xFFFF) {
                   if (MessageBox(hDlg, (LPSTR) FileExists, (LPSTR) FileName,
                      MB_OKCANCEL | MB_ICONQUESTION) == IDCANCEL)
                      return (TRUE);
                 }
                hFile = OpenFile(str, &lpOFSTRUCT, OF_CREATE);
                if (hFile == 0xFFFF) {
                   MessageBox(hDlg, (LPSTR) UnableOpenEr,
                      (LPSTR) NULL, MB_OK | MB_ICONQUESTION);
                   return (TRUE);
                 }
                close (hFile);      /* leave file closed and reopen when needed*/

                _fstrcpy(FileStuff.Name, FileName);
                _fstrcpy(FileStuff.Path, DefPath);
```


167

```
                memcpy(&FileStuff.lpOFSTRUCT, &lpOFSTRUCT, sizeof(OFSTRUCT));

                nTotal = 0;
                szName[0]     = 0;
                szClass[0]       = 0;
                szOwner[0]       = 0;
                szClassifi[0]    = 0;
                szBuilder[0] = 0;
                szDelivery[0]    = 0;
                szRoute[0]       = 0;
                szOther[0]       = 0;

                fModified = FALSE;
                WriteText(hDlg);
                EndDialog(hDlg, NULL);

               return (TRUE);

            case IDCANCEL:
                EndDialog(hDlg, NULL);
                stpcpy(ShipType, OldType);
                return (TRUE);

            default:
                return (TRUE);
         }

      case WM_INITDIALOG:
         _fstrcpy(str, DefPath);
         _fstrcat(str, DefSpec);
         _fstrcpy(PathName, DefPath);
         /* Update Path list box */
         DlgDirList(hDlg, str, SaveAs_DirLb, SaveAs_DirEdit, 0xC010);
         SetDlgItemText(hDlg, SaveAs_FileEdit, FileStuff.Name);
         SetFocus(GetDlgItem(hDlg, SaveAs_FileEdit));
         return (FALSE);

      case WM_CLOSE:
         PostMessage(hDlg, WM_COMMAND, IDCANCEL, 0L);
         break;
   }
return (NULL);
}

/****************************************************************************

   FUNCTION:     SelectLayout(HWND, unsigned, WORD, LONG)

   PURPOSE:      While users select 'File_New', this function pop up a dialog
                 box to choose a ship layout or enter a user bitmap name.

   MESSAGES: WM_COMMAND, WM_INITDIALOG

****************************************************************************/
BOOL FAR PASCAL SelectLayout(hDlg, message, wParam, lParam)
HWND      hDlg;
unsigned  message;
WORD      wParam;
LONG      lParam;
{
   FARPROC lpProcAbout;
   char BitmapName[Max_FilePath];
   BOOL chk;


   switch (message){
      case WM_COMMAND:
         switch (wParam){
          // case 151:
            case 152:
            case 153:
            case 154:
```

```
            case 155:
               CheckRadioButton(hDlg, 152, 155, wParam);
               break;

            case IDOK:
               userBitmap = 0;
               stpcpy(OldType, ShipType);
            // if(IsDlgButtonChecked(hDlg,151) > 0) stpcpy(ShipType, "SHIPTYPE1");
               if(IsDlgButtonChecked(hDlg,152) > 0) stpcpy(ShipType, "SHIPTYPE2");
               if(IsDlgButtonChecked(hDlg,153) > 0) stpcpy(ShipType, "SHIPTYPE3");
               if(IsDlgButtonChecked(hDlg,154) > 0) stpcpy(ShipType, "SHIPTYPE4");
               if(IsDlgButtonChecked(hDlg,155) > 0) {
                 chk = GetDlgItemText(hDlg, 156, (LPSTR) ShipType, 40);
                  if(chk == 0){
                     MessageBox(hDlg, "Invalid Data Input", NULL,
                                MB_OK | MB_ICONEXCLAMATION);
                  break;
                     };
                 userBitmap = 1;
                 };
               lpProcAbout = MakeProcInstance(NewFile, hInst);
               DialogBox(hInst, "FileNew", hWnd, lpProcAbout);
               FreeProcInstance(lpProcAbout);

               EndDialog(hDlg,NULL);
               InvalidateRect(hWnd, NULL, 1);
               PostMessage(hWnd, WM_PAINT, NULL, NULL);
               break;

            case IDCANCEL:
               EndDialog(hDlg,NULL);
               return (TRUE);
         }
         break;

      case WM_INITDIALOG:
         CheckRadioButton(hDlg, 152, 155, 155);
         return (TRUE);

      case WM_PAINT:
         myPaint(hDlg, "SELECTSHIP");
         return (TRUE);

      case WM_CLOSE:
         PostMessage(hDlg, WM_COMMAND, IDCANCEL, 0L);
         break;
      }
   return (FALSE);
}
```

A.8 FILEFMT.C

```
/***************************************************************************

   Module:      FileFmt.C

   PURPOSE:     Writes the different file formats.

   FUNCTIONS:

   EXPORTED:    WriteText(hDlg, str)
                ReadNativeFormat()
   LOCAL:    WriteParam()
                ReadParam()
                ReadTxt()

***************************************************************************/
#define fFileStuff  TRUE

#include <string.h>
#include <windows.h>
#include <io.h>
#include <stdio.h>
#include <math.h>
#include "Repair.h"

extern BOOL  userBitmap;
extern int   fModified, nTotal, nX[MAX][3], nY[MAX][3], nClss[MAX];
extern char ShipType[20];
extern char  szName[MAXG], szClass[MAXG], szOwner[MAXG], szClassifi[MAXG],
             szBuilder[MAXG], szDelivery[MAXG], szRoute[MAXG], szOther[MAXG],
             szLocation[MAX][MAXL], szDate[MAX][MAXD], szLength[MAX][MAXLE],
             szStatus[MAX][MAXS], szComment[MAX][MAXC], szCrkType[MAX][MAXCT];

extern char far FileSignature[];
const char *strvoid ="Void";
const char *IfUseBitmap ="\nUse Bitmap?: ";
const char *GeneralLayout ="General Layout: ";
const char *TotalCrackNum ="\nTotal crack number: ";
const char *CrackTxt ="Crack#";
const char *LocaTxt ="Location#";
const char *DateTxt ="Date#";
const char *LengTxt ="Length#";
const char *StatTxt ="Status#";
const char *CommTxt ="Comment#";
const char *CrkTTxt ="Graph#";

extern BOOL MyRead (int, LPSTR, DWORD);

/***************************************************************************

   FUNCTION:   WriteParam(HWND, LPSTR)

   PURPOSE:    Writes parameter to output buffer.

***************************************************************************/
void WriteParam(Diskbuf, param)
LPSTR Diskbuf;
char *param;
{
   if (param[0] == 0){
   _fstrcat(Diskbuf, strvoid);
   } else {
   _fstrcat(Diskbuf, param);
   }
_fstrcat(Diskbuf, "\n");
}

/***************************************************************************

   FUNCTION:   ReadParam()
```

```
   PURPOSE:    Finds token in the buffer and takes the next word for
            the return parameter.

****************************************************************************/
void ReadParam(pbuffer, ptoken, pdest)
LPSTR pbuffer;
char *ptoken, *pdest;
{
   char *pNext;

   pNext = strstr(pbuffer, ptoken);
   if (pNext != NULL)
   {
      pNext += strlen(ptoken);
      sscanf(pNext,"%s",pdest);
      if (0 == strcmpi(pdest,strvoid)) pdest[0] = 0;
   }
   else
   {
      pdest[0] = 0;
   }
}

/****************************************************************************

   FUNCTION:   ReadTxt()

   PURPOSE:    Finds token in the buffer and takes the next text for
            the return parameter. (Read until "\n")

****************************************************************************/
void ReadTxt(pbuffer, ptoken, pdest)
LPSTR pbuffer;
char *ptoken, *pdest;
{
   int  i;
   char c;
   char *pNext;

   pNext = strstr(pbuffer, ptoken);
   if (pNext != NULL)
   {
      i=0;
      pNext += strlen(ptoken);
      sscanf(pNext, "%c", &c);
      while((c != 0x0a) && (c != 0x0d))
      {
         pdest[i] = c;
         i++;
         pNext ++;
         sscanf(pNext, "%c", &c);
      }
      pdest[i] = 0;
      if (0 == strcmpi(pdest,strvoid)) pdest[0] = 0;
   }
   else
   {
      pdest[0] = 0;
   }
}

/****************************************************************************

   FUNCTION:   WriteText(HWND)

   PURPOSE:    Writes text into a file.

****************************************************************************/
void FAR WriteText(hDlg)
HWND        hDlg;
{
   HANDLE  hFileNew;
```

```
char tempstr[80], *pEnd;
char *CrackTxtNum;
int count, hDiskbuf, i;
DWORD dwFileLength;
LPSTR Diskbuf;
OFSTRUCT lpOFSTRUCT;

hFileNew = OpenFile(NULL, &FileStuff.lpOFSTRUCT,
            OF_REOPEN | OF_CREATE | OF_PROMPT | OF_WRITE);
if (hFileNew == 0xFFFF){
   sprintf(tempstr, "Cannot save file - %s",FileStuff.Name);
   MessageBox(hDlg, tempstr, NULL, MB_OK | MB_ICONEXCLAMATION);
   return;
   }

dwFileLength=(DWORD) (116+200+ nTotal*(95+MAXL+MAXD+MAXLE+MAXS+MAXC+MAXCT));;
hDiskbuf = LocalAlloc(LMEM_MOVEABLE | LMEM_ZEROINIT, dwFileLength);
if (hDiskbuf == 0) {
   MessageBox(hDlg, "Not enough memory to save file!",
            NULL, MB_OK | MB_ICONEXCLAMATION);
   return;
   }

Diskbuf = LocalLock(hDiskbuf);

/* Begin to write data */
_fstrcpy(Diskbuf, FileSignature);
_fstrcat(Diskbuf, "\r\n\r\n");
_fstrcat(Diskbuf, "Name: ");
if(szName[0]==0) sprintf(Diskbuf+strlen(Diskbuf),"%s\n",strvoid);
else sprintf(Diskbuf+strlen(Diskbuf), "%s\n", szName);

_fstrcat(Diskbuf, "Class: ");
if(szClass[0]==0) sprintf(Diskbuf+strlen(Diskbuf),"%s\n",strvoid);
else sprintf(Diskbuf+strlen(Diskbuf), "%s\n", szClass);

_fstrcat(Diskbuf, "Owner: ");
if(szOwner[0]==0) sprintf(Diskbuf+strlen(Diskbuf),"%s\n",strvoid);
else sprintf(Diskbuf+strlen(Diskbuf), "%s\n", szOwner);

_fstrcat(Diskbuf, "Classification: ");
if(szClassifi[0]==0) sprintf(Diskbuf+strlen(Diskbuf),"%s\n",strvoid);
else sprintf(Diskbuf+strlen(Diskbuf), "%s\n", szClassifi);

_fstrcat(Diskbuf, "Builder: ");
if(szBuilder[0]==0) sprintf(Diskbuf+strlen(Diskbuf),"%s\n",strvoid);
else sprintf(Diskbuf+strlen(Diskbuf), "%s\n", szBuilder);

_fstrcat(Diskbuf, "Delivery: ");
if(szDelivery[0]==0) sprintf(Diskbuf+strlen(Diskbuf),"%s\n",strvoid);
else sprintf(Diskbuf+strlen(Diskbuf), "%s\n", szDelivery);

_fstrcat(Diskbuf, "Route: ");
if(szRoute[0]==0) sprintf(Diskbuf+strlen(Diskbuf),"%s\n",strvoid);
else sprintf(Diskbuf+strlen(Diskbuf), "%s\n", szRoute);

_fstrcat(Diskbuf, "Others: ");
if(szOther[0]==0) sprintf(Diskbuf+strlen(Diskbuf),"%s\n",strvoid);
else sprintf(Diskbuf+strlen(Diskbuf), "%s\n", szOther);


_fstrcat(Diskbuf, IfUseBitmap);
sprintf(Diskbuf+strlen(Diskbuf), "%d\n", userBitmap);
_fstrcat(Diskbuf, GeneralLayout);
sprintf(Diskbuf+strlen(Diskbuf), "%s\n", ShipType);

_fstrcat(Diskbuf, TotalCrackNum);
sprintf(Diskbuf+strlen(Diskbuf), "%d\n", nTotal);

for(i=1; i<=nTotal; i++)
{
   sprintf(Diskbuf+strlen(Diskbuf), "Crack#%d: ", i);
   sprintf(Diskbuf+strlen(Diskbuf), "%d %d %d %d %d %d %d\n",
```


172

```
        nX[i][0], nY[i][0], nX[i][1], nY[i][1], nX[i][2], nY[i][2],
        nClss[i]/*, nFig[i]*/);

      sprintf(Diskbuf+strlen(Diskbuf), "Location#%d: ", i);
      if(szLocation[i][0]==0) sprintf(Diskbuf+strlen(Diskbuf),"%s\n",strvoid);
      else sprintf(Diskbuf+strlen(Diskbuf), "%s\n", szLocation[i]);

      sprintf(Diskbuf+strlen(Diskbuf), "Date#%d: ", i);
      if(szDate[i][0]==0) sprintf(Diskbuf+strlen(Diskbuf),"%s\n",strvoid);
      else sprintf(Diskbuf+strlen(Diskbuf), "%s\n", szDate[i]);

      sprintf(Diskbuf+strlen(Diskbuf), "Length#%d: ", i);
      if(szLength[i][0]==0) sprintf(Diskbuf+strlen(Diskbuf),"%s\n",strvoid);
      else sprintf(Diskbuf+strlen(Diskbuf), "%s\n", szLength[i]);

      sprintf(Diskbuf+strlen(Diskbuf), "Status#%d: ", i);
      if(szStatus[i][0]==0) sprintf(Diskbuf+strlen(Diskbuf),"%s\n",strvoid);
      else sprintf(Diskbuf+strlen(Diskbuf), "%s\n", szStatus[i]);

      sprintf(Diskbuf+strlen(Diskbuf), "Comment#%d: ", i);
      if(szComment[i][0]==0) sprintf(Diskbuf+strlen(Diskbuf),"%s\n",strvoid);
      else sprintf(Diskbuf+strlen(Diskbuf), "%s\n\n", szComment[i]);

      sprintf(Diskbuf+strlen(Diskbuf), "Graph#%d: ", i);
      if(szCrkType[i][0]==0) sprintf(Diskbuf+strlen(Diskbuf),"%s\n\n",strvoid);
      else sprintf(Diskbuf+strlen(Diskbuf), "%s\n\n", szCrkType[i]);
      };
   fModified = FALSE;

   count = _fstrlen(Diskbuf);
   if(_lwrite(hFileNew, Diskbuf, count) <= 0){
      sprintf(tempstr, "Cannot write file - %s",FileStuff.Name);
      MessageBox(hDlg, tempstr, NULL, MB_OK | MB_ICONEXCLAMATION);
      }

   _lclose(hFileNew);
   LocalUnlock(hDiskbuf);
   LocalFree(hDiskbuf);
}

/****************************************************************************

   FUNCTION:   ReadNativeFormat(HWND)

   PURPOSE:    Reads text file written by this program

****************************************************************************/
void FAR ReadNativeFormat(hDlg)
HWND       hDlg;
{
   HANDLE  hFile;
   char tempstr[20], tstr0[20], tstr1[20], tstr2[20], tstr3[20], tstr4[80];
   char *CrackTxtNum;
   int hDiskbuf, i;
   LPSTR Diskbuf;
   OFSTRUCT lpOFSTRUCT;


   hFile = OpenFile(NULL, &FileStuff.lpOFSTRUCT, OF_REOPEN | OF_READ);
   if (hFile == 0xFFFF){
      MessageBox(hDlg, "Error reopening file!", NULL,
                 MB_OK | MB_ICONEXCLAMATION);
      return;
      }

   hDiskbuf = LocalAlloc(LMEM_MOVEABLE | LMEM_ZEROINIT, filelength (hFile));
   if (hDiskbuf == 0) {
      MessageBox(hDlg, "Not enough memory to read file!",
               NULL, MB_OK | MB_ICONEXCLAMATION);
      return;
      }
```

```
Diskbuf = LocalLock(hDiskbuf);
if (!MyRead (hFile, Diskbuf, filelength (hFile)))
   {
   MessageBox(hDlg, "Read into buffer wrong!",
            NULL, MB_OK | MB_ICONEXCLAMATION);
   LocalUnlock (hDiskbuf);
   LocalFree   (hDiskbuf);
   return;
   }

/* Read data */
ReadTxt(Diskbuf, "Name: ",            szName);
ReadTxt(Diskbuf, "Class: ",         szClass);
ReadTxt(Diskbuf, "Owner: ",         szOwner);
ReadTxt(Diskbuf, "Classification: ", szClassifi);
ReadTxt(Diskbuf, "Builder: ",          szBuilder);
ReadTxt(Diskbuf, "Delivery: ",      szDelivery);
ReadTxt(Diskbuf, "Route: ",         szRoute);
ReadTxt(Diskbuf, "Others: ",           szOther);

ReadParam(Diskbuf, IfUseBitmap, tempstr);
userBitmap = atoi(tempstr);
ReadParam(Diskbuf, GeneralLayout, ShipType);
ReadParam(Diskbuf, TotalCrackNum, tempstr);
nTotal = atoi(tempstr);

for(i=1; i<=nTotal; i++)
   {

   /* Read Crack Coordinates */
   _fstrcpy(CrackTxtNum, CrackTxt);
   itoa(i, tstr0, 10);

   _fstrcat(CrackTxtNum, tstr0);
   _fstrcat(CrackTxtNum, ": ");
   ReadParam(Diskbuf, CrackTxtNum, tstr1);
   nX[i][0] = atoi(tstr1);                      /* x1 */

   _fstrcat(CrackTxtNum, tstr1);
   _fstrcat(CrackTxtNum, " ");
   ReadParam(Diskbuf, CrackTxtNum, tstr2);
   nY[i][0] = atoi(tstr2);                      /* y1 */

   _fstrcat(CrackTxtNum, tstr2);
   _fstrcat(CrackTxtNum, " ");
   ReadParam(Diskbuf, CrackTxtNum, tstr1);
   nX[i][1] = atoi(tstr1);                      /* x2 */

   _fstrcat(CrackTxtNum, tstr1);
   _fstrcat(CrackTxtNum, " ");
   ReadParam(Diskbuf, CrackTxtNum, tstr2);
   nY[i][1] = atoi(tstr2);                      /* y2 */

   _fstrcat(CrackTxtNum, tstr2);
   _fstrcat(CrackTxtNum, " ");
   ReadParam(Diskbuf, CrackTxtNum, tstr1);
   nX[i][2] = atoi(tstr1);                      /* x3 */

   _fstrcat(CrackTxtNum, tstr1);
   _fstrcat(CrackTxtNum, " ");
   ReadParam(Diskbuf, CrackTxtNum, tstr2);
   nY[i][2] = atoi(tstr2);                      /* y3 */

   _fstrcat(CrackTxtNum, tstr2);
   _fstrcat(CrackTxtNum, " ");
   ReadParam(Diskbuf, CrackTxtNum, tstr3);
   nClss[i] = atoi(tstr3);                      /* Crack Class */

   /* Read Location, Date, Length,... */
   _fstrcpy(CrackTxtNum, LocaTxt);
   itoa(i, tstr0, 10);
   _fstrcat(CrackTxtNum, tstr0);
   _fstrcat(CrackTxtNum, ": ");
```

```
 ReadTxt(Diskbuf, CrackTxtNum, szLocation[i]);

 _fstrcpy(CrackTxtNum, DateTxt);
 itoa(i, tstr0, 10);
 _fstrcat(CrackTxtNum, tstr0);
 _fstrcat(CrackTxtNum, ": ");
 ReadTxt(Diskbuf, CrackTxtNum, szDate[i]);

 _fstrcpy(CrackTxtNum, LengTxt);
 itoa(i, tstr0, 10);
 _fstrcat(CrackTxtNum, tstr0);
 _fstrcat(CrackTxtNum, ": ");
 ReadTxt(Diskbuf, CrackTxtNum, szLength[i]);

 _fstrcpy(CrackTxtNum, StatTxt);
 itoa(i, tstr0, 10);
 _fstrcat(CrackTxtNum, tstr0);
 _fstrcat(CrackTxtNum, ": ");
 ReadTxt(Diskbuf, CrackTxtNum, szStatus[i]);

 _fstrcpy(CrackTxtNum, CommTxt);
 itoa(i, tstr0, 10);
 _fstrcat(CrackTxtNum, tstr0);
 _fstrcat(CrackTxtNum, ": ");
 ReadTxt(Diskbuf, CrackTxtNum, szComment[i]);

 _fstrcpy(CrackTxtNum, CrkTTxt);
 itoa(i, tstr0, 10);
 _fstrcat(CrackTxtNum, tstr0);
 _fstrcat(CrackTxtNum, ": ");
 ReadParam(Diskbuf, CrackTxtNum, szCrkType[i]);
 };

 fModified = FALSE;

 _lclose(hFile);
 LocalUnlock(hDiskbuf);
 LocalFree(hDiskbuf);
}
/****************************************************************************

   FUNCTION:   ReadRepairData(HWND)

   PURPOSE:    Reads text file which store repair SN data and stress ratios.

****************************************************************************/
void FAR ReadRepairData(hDlg, iDtl, iRpr, fA, fm, fnewA, fnewm, fratio)
HWND     hDlg;
int   iDtl, iRpr;
float    *fA, *fm, *fnewA, *fnewm, *fratio;
{
   HANDLE  hFile;
   char tempstr[20], tstr0[20], tstr1[20], tstr2[20];
   char *CrackTxtNum;
   int hDiskbuf;
   LPSTR Diskbuf;
   OFSTRUCT lpOFSTRUCT;

   hFile = OpenFile("REPAIR.DAT", &FileStuff.lpOFSTRUCT, OF_READ);
   if (hFile == 0xFFFF){
      MessageBox(hDlg, "Error reopening file!", NULL,
                  MB_OK | MB_ICONEXCLAMATION);
      return;
      }

   hDiskbuf = LocalAlloc(LMEM_MOVEABLE | LMEM_ZEROINIT, filelength (hFile));
   if (hDiskbuf == 0) {
      MessageBox(hDlg, "Not enough memory to read file!",
               NULL, MB_OK | MB_ICONEXCLAMATION);
      return;
      }
```

```
    Diskbuf = LocalLock(hDiskbuf);
    if (!MyRead (hFile, Diskbuf, filelength (hFile)))
       {
       MessageBox(hDlg, "Read into buffer wrong!",
                  NULL, MB_OK | MB_ICONEXCLAMATION);
       LocalUnlock (hDiskbuf);
       LocalFree   (hDiskbuf);
       return;
       }

    /* Read data */
    _fstrcpy(CrackTxtNum, "Detail#");
    itoa(iDtl, tstr0, 10);
    _fstrcat(CrackTxtNum, tstr0);
    _fstrcat(CrackTxtNum, "Repair#");
    itoa(iRpr, tstr0, 10);
    _fstrcat(CrackTxtNum, tstr0);
    _fstrcat(CrackTxtNum, ": ");

    ReadParam(Diskbuf, CrackTxtNum, tstr1);
    *fA = atof(tstr1);                        /* SN A */

    _fstrcat(CrackTxtNum, tstr1);
    _fstrcat(CrackTxtNum, " ");
    ReadParam(Diskbuf, CrackTxtNum, tstr2);
    *fm = atof(tstr2);                        /* SN m */

    _fstrcat(CrackTxtNum, tstr2);
    _fstrcat(CrackTxtNum, " ");
    ReadParam(Diskbuf, CrackTxtNum, tstr1);
    *fnewA = atof(tstr1);                        /* SN A' */

    _fstrcat(CrackTxtNum, tstr1);
    _fstrcat(CrackTxtNum, " ");
    ReadParam(Diskbuf, CrackTxtNum, tstr2);
    *fnewm = atof(tstr2);                        /* SN m' */

    _fstrcat(CrackTxtNum, tstr2);
    _fstrcat(CrackTxtNum, " ");
    ReadParam(Diskbuf, CrackTxtNum, tstr1);
    *fratio = atof(tstr1);                        /* Stress Ratio */

    _lclose(hFile);
    LocalUnlock(hDiskbuf);
    LocalFree(hDiskbuf);
}
```

A.9 FILEUTIL.C

```
/******************************************************************************

   Module:     FileUtil.c

   PURPOSE:    Common procedures for file commands.

   FUNCTIONS:
      EXPORTED:
         UpdateListBox()
         SeparateFile()
         ChangeDefExt()
         UpTreePath()
         VerifySig()
         QuerySaveFile()

******************************************************************************/
#include <string.h>
#include <windows.h>
#include <io.h>
#include <stdio.h>
#include "repair.h"

extern int fModified;
extern VOID FAR          SaveFile(HWND);
/******************************************************************************

   FUNCTION:   UpdateListBox(HWND, LPSTR, LPSTR, LPSTR)

   PURPOSE:    Fills in the Files list box.

******************************************************************************/
void FAR UpdateListBox(hDlg, lPath, lName, lstrret)
HWND       hDlg;
LPSTR      lPath, lName, lstrret;
{
   char lstring[Max_FilePath + 14];

   lstrcpy(lstring, lPath);
   lstrcat(lstring, lName);/* create file search patern */
   /* The following will update the file list box and the path edit box */
   /* This call kills string !*/
   DlgDirList(hDlg, lstring, IDC_LISTBOX, IDC_PATH, 0);
   SetDlgItemText(hDlg, IDC_EDIT, lName);
   /* the following will update the path list box */
   lstrcpy(lstring, lPath);
   lstrcat(lstring, lName);/* create file search patern */
   DlgDirList(hDlg, lstring, IDC_PATHBOX, 0, 0xC010);
   lstrcpy(lstrret, lPath);
   lstrcat(lstrret, lName);/* create file search patern */
}

/******************************************************************************

   FUNCTION:   SeparateFile(LPSTR, LPSTR, LPSTR)

   PURPOSE:    Breaks up path\filename.

******************************************************************************/
void FAR SeparateFile(lpDestPath, lpDestFileName, lpSrcFileName)
LPSTR      lpDestPath, lpDestFileName, lpSrcFileName;
{
   LPSTR lpTmp;
   char cTmp;

   lpTmp = lpSrcFileName + (long) lstrlen(lpSrcFileName);

   while (*lpTmp !=':' && *lpTmp != '\\' && lpTmp > lpSrcFileName)
      lpTmp = AnsiPrev(lpSrcFileName,lpTmp);
```

```
    if (*lpTmp != ':' && *lpTmp != '\\') {
        lstrcpy(lpDestFileName, lpSrcFileName);
        lpDestPath[0] = 0;
        return;
    }
    lstrcpy(lpDestFileName, lpTmp + 1);
    cTmp = *(lpTmp+1);
    lstrcpy(lpDestPath, lpSrcFileName);
    *(lpTmp + 1) = cTmp;
    lpDestPath[(lpTmp - lpSrcFileName) + 1] = 0;

}
/****************************************************************************

    FUNCTION:   ChangeDefExt(PSTR, PSTR)

    PURPOSE:    Detects an extension of a file (or wildcard), and if present
              copies the extension to the string.

****************************************************************************/
void FAR ChangeDefExt(Ext, Name)
LPSTR Ext, Name;
{
    LPSTR pTptr;

    pTptr = Name;
    while (*pTptr && *pTptr != '.')
        pTptr++;
    if (!*pTptr) _fstrcat(pTptr,Ext);/* no extention was given so add one */

}
/****************************************************************************

    FUNCTION:   UpTreePath(LPSTR)

    PURPOSE:    To take the current directory search path such as
              '\root\dir\'  and go up the directory tree by
              removing 'dir\' to form '\root\'

****************************************************************************/
void FAR UpTreePath (lFileSpec)
LPSTR  lFileSpec;
{
    int index;

    index = lstrlen(lFileSpec) - 2;/* we know the last byte will be a '\' */
    while ((index != 0) && (lFileSpec[index] != '\\')){
        index -= 1;
        }
    lFileSpec[++index] = 0;
}
/****************************************************************************

    FUNCTION:   VerifySig(int, LPSTR)

    PURPOSE:    To verify that the file opened is for this program.

****************************************************************************/
int FAR VerifySig (hFile, lSignature)
int hFile;
LPSTR  lSignature;
{
    char signature[80];
    int count, hDiskbuf;
    PSTR Diskbuf;

    hDiskbuf = LocalAlloc(LMEM_MOVEABLE | LMEM_ZEROINIT, 512);
    if (!hDiskbuf) return (FALSE);/* can't get memory ?*/

    /* make sure we are at the start of file */
```

```
    if ( _llseek(hFile, 0, 0)) return (FALSE);
    Diskbuf = LocalLock(hDiskbuf);
    count = _lread(hFile, Diskbuf, 512);
    if (count <= 0) goto returnfalse;
    if (0 >= sscanf(Diskbuf,"%[^\n\r]",signature)) goto returnfalse;
    if(_fstrcmp(lSignature, signature) != 0) goto returnfalse;

    LocalUnlock(hDiskbuf);
    LocalDiscard(hDiskbuf);
    return (TRUE);

    returnfalse:
    LocalUnlock(hDiskbuf);
    LocalDiscard(hDiskbuf);
    return FALSE;
}

/****************************************************************************

    FUNCTION:   QuerySaveFile()

    PURPOSE:    To allow user to save a file befor exiting.

****************************************************************************/
VOID FAR QuerySaveFile (hDlg)
HWND        hDlg;
{
    if (fModified){
        if (MessageBox(hDlg, "Save file?", "Last Chance!", MB_YESNO |
            MB_ICONQUESTION) == IDNO)   return;

            SaveFile(hDlg);
    }
}
```


179

A.10 MAIN.C

```
/****************************************************************************

   PROGRAM:    Main.c

   PURPOSE:    "Repair Management System for Tankers" is a software that
         help repair engineers to maintain their ships.

         This module "Main.c" does initialization and created the Main
         Window.

   FUNCTIONS:

   WinMain() - Calls initialization function, processes message loop.
   InitApplication() - Initializes window data and registers window class.
   InitInstance() - Saves instance handle and creates main window.
   About() - Processes messages for "About" dialog box.
   MakeHelpPathName() - Derives path name of help file.
   Engrayed() - Engray menu items before a file is selected.

****************************************************************************/
#define  fFileStuff TRUE

#include <windows.h>
#include "repair.h"

HWND      hWnd;                 /* Handle to main window */
HANDLE    hInst;                /* Handle to instance data*/
BOOL      bHelp = FALSE;        /* Help mode flag; TRUE = "ON"*/
HCURSOR   hHelpCursor;          /* Cursor displayed when in help mode*/
char      szHelpFileName[EXE_NAME_MAX_SIZE+1];    /* Help file name*/
HANDLE    hAccTable;            /* handle to accelerator table */

extern long FAR PASCAL MainWndProc(HWND, unsigned, WORD, LONG);
BOOL FAR PASCAL About(HWND, unsigned, WORD, LONG);
void MakeHelpPathName(char*);  /* Function deriving help file path */
void EnableGrayed(HWND);

/****************************************************************************

   FUNCTION:   WinMain(HANDLE, HANDLE, LPSTR, int)

   PURPOSE:    Calls initialization function, processes message loop.

****************************************************************************/
int PASCAL WinMain(hInstance, hPrevInstance, lpCmdLine, nCmdShow)
HANDLE hInstance;
HANDLE hPrevInstance;
LPSTR  lpCmdLine;
int    nCmdShow;
{
   MSG msg;

   if (!hPrevInstance)
       if (!InitApplication(hInstance))
       return (FALSE);

   if (!InitInstance(hInstance, nCmdShow))
       return (FALSE);

   while (GetMessage(&msg, NULL, NULL, NULL)) {
      /* Only translate message if it is not an accelerator message */
      if (!TranslateAccelerator(hWnd, hAccTable, &msg)) {
          TranslateMessage(&msg);
          DispatchMessage(&msg);
       }
   }
   return (msg.wParam);
}
```

```
/****************************************************************************

   FUNCTION:    InitApplication(HANDLE)

   PURPOSE:     Initializes window data and registers window class.

   RETURNS:     Status of RegisterClass() call.

****************************************************************************/
BOOL InitApplication(hInstance)
HANDLE hInstance;
{
   WNDCLASS wc;

   wc.style = NULL; /*CS_HREDRAW | CS_VREDRAW | CS_BYTEALIGNWINDOW;*/
   wc.lpfnWndProc = MainWndProc;
   wc.cbClsExtra = 0;
   wc.cbWndExtra = 0;
   wc.hInstance = hInstance;
   wc.hIcon = LoadIcon(hInstance, "TANKER0");
   wc.hCursor = NULL; //LoadCursor(NULL, IDC_ARROW);
   wc.hbrBackground = GetStockObject(WHITE_BRUSH);
   wc.lpszMenuName ="MainMenu";
   wc.lpszClassName = "RepairWClass";

   return (RegisterClass(&wc));
}


/****************************************************************************

   FUNCTION:    InitInstance(HANDLE, int)

   PURPOSE:     Saves instance handle in global variable and creates main
                window.

   RETURNS:     Status of CreateWindow() call.

****************************************************************************/
BOOL InitInstance(hInstance, nCmdShow)
HANDLE hInstance;
int    nCmdShow;
{
   OFSTRUCT OFHelp;
   hInst = hInstance;

   hAccTable = LoadAccelerators(hInst, "HelpexAcc");

   hWnd = CreateWindow(
       "RepairWClass",
       "Repair Management System",
       WS_OVERLAPPEDWINDOW,
       CW_USEDEFAULT,
       CW_USEDEFAULT,
       CW_USEDEFAULT,
       CW_USEDEFAULT,
       NULL,
       NULL,
       hInstance,
       NULL
       );

   if (!hWnd)
       return (FALSE);

   ShowWindow(hWnd, nCmdShow);
   UpdateWindow(hWnd);

   EnableGrayed(hWnd);

   MakeHelpPathName(szHelpFileName);
   if(OpenFile(szHelpFileName, &OFHelp, OF_EXIST) < 1){  /* No help file */
```

```
        EnableMenuItem(GetMenu(hWnd), 301, MF_GRAYED);}      /* So gray it.  */

    hHelpCursor = LoadCursor(hInst,"HelpCursor");

    return (TRUE);
}

/****************************************************************************

    FUNCTION:   About(HWND, unsigned, WORD, LONG)

    PURPOSE:    Processes messages for "About" dialog box

    MESSAGES:

        WM_INITDIALOG - Initialize dialog box
        WM_COMMAND- Input received

****************************************************************************/

BOOL FAR PASCAL About(hDlg, message, wParam, lParam)
HWND        hDlg;
unsigned    message;
WORD        wParam;
LONG        lParam;
{
    switch (message) {
        case WM_INITDIALOG:
            return (TRUE);

        case WM_COMMAND:
            if (wParam == IDOK) {
                EndDialog(hDlg, TRUE);
                return (TRUE);
            }
            break;
    }
    return (FALSE);
}

/****************************************************************************

    FUNCTION:   MakeHelpPathName

    PURPOSE:    This code assumes that the .HLP help file is in the same
            directory as the Repair executable. This function derives
                the full path name of the help file from the path of the
                executable.

****************************************************************************/

void MakeHelpPathName(szFileName)
char * szFileName;
{
    char *  pcFileName;
    int     nFileNameLen;

    nFileNameLen = GetModuleFileName(hInst,szFileName,EXE_NAME_MAX_SIZE);
    pcFileName = szFileName + nFileNameLen;

    while (pcFileName > szFileName) {
        if (*pcFileName == '\\' || *pcFileName == ':') {
            *(++pcFileName) = '\0';
            break;
        }
    nFileNameLen--;
    pcFileName--;
    }

    if ((nFileNameLen+13) < EXE_NAME_MAX_SIZE)
        { lstrcat(szFileName, "repair1.hlp"); }
    else { lstrcat(szFileName, "?");}
```

```
   return;
}

/****************************************************************************

   FUNCTION:    EnableGrayed(HWND)

   PURPOSE:     Enables grayed (disabled) commands when a file is open.

****************************************************************************/
void EnableGrayed(hWnd)
HWND        hWnd;
{
   HMENU     hMenu;
   int    state;

   hMenu = GetMenu(hWnd);
   state = (FileStuff.Name[0] == 0) ? MF_GRAYED: MF_ENABLED;
   EnableMenuItem(hMenu, 102, state | MF_BYCOMMAND); /* Save */
   EnableMenuItem(hMenu, 103, state | MF_BYCOMMAND); /* SaveAS */
   EnableMenuItem(hMenu, 1, state | MF_BYPOSITION); /* Ship */
   EnableMenuItem(hMenu, 2, state | MF_BYPOSITION); /* Edit */
   DrawMenuBar(hWnd);  /* to reflect the current sate of menu items */
}
```

A.11 MAINWND.C

```
/****************************************************************************

    Module:    MainWnd.C

    PURPOSE:    Processes window messages.

    FUNCTIONS:  DrawCrack(hWnd, int, int, BOOL)
                DrawIndex(HWND, int, int, BOOL)
                DrawAllCrack(HWND)
                EraseIndexSet(HWND, int)
                EraseCrackSet(HWND, int)
                MainWndProc(HWND, unsigned, WORD, LONG)

****************************************************************************/
#define  fFileStuff TRUE

#include <windows.h>
#include <string.h>
#include "repair.h"

intnTotal, nSelect, nAddCrack, ntempFig;
intnX[MAX][3], nY[MAX][3], nClss[MAX]/*, nFig[MAX]*/;
char   szName[MAXG], szClass[MAXG], szOwner[MAXG], szClassifi[MAXG],
       szBuilder[MAXG], szDelivery[MAXG], szRoute[MAXG], szOther[MAXG],
       szLocation[MAX][MAXL], szDate[MAX][MAXD], szLength[MAX][MAXLE],
       szStatus[MAX][MAXS], szComment[MAX][MAXC], szCrkType[MAX][MAXCT];
extern char     ShipType[20], CrackType[MAXCT];
extern char     szHelpFileName[];       /* Help file name */
extern HANDLE   hInst;                  /* Handle to instance data */
extern BOOL     bHelp; /* = FALSE */    /* Help mode flag; TRUE = "ON"*/
extern HCURSOR  hHelpCursor;            /* Cursor displayed when in help mode*/

extern BOOL                 userBitmap;
extern BOOL FAR PASCAL      userPaint(HWND, char*);

extern BOOL FAR PASCAL      About(HWND, unsigned, WORD, LONG);
extern BOOL FAR PASCAL      SaveAs(HWND, unsigned, WORD, LONG);
extern BOOL FAR PASCAL      OpenSource(HWND, unsigned, WORD, LONG);
extern BOOL FAR PASCAL      NewFile(HWND, unsigned, WORD, LONG);
extern BOOL FAR PASCAL      SelectLayout(HWND, unsigned, WORD, LONG);
extern VOID FAR             SaveFile(HWND);
extern BOOL FAR PASCAL      Analysis(HWND, unsigned, WORD, LONG);
extern BOOL FAR PASCAL      ShipGeneral(HWND, unsigned, WORD, LONG);
extern BOOL FAR PASCAL      AddRecord(HWND, unsigned, WORD, LONG);
extern BOOL FAR PASCAL      EditRecord(HWND, unsigned, WORD, LONG);
extern BOOL FAR PASCAL      DeleteCrk(HWND, unsigned, WORD, LONG);
extern BOOL FAR PASCAL      myPaint(HWND, char*);
extern void              EnableGrayed(HWND);
extern VOID FAR             QuerySaveFile (HWND);

/****************************************************************************

    FUNCTION:   DrawCrack(hWnd, int, int, BOOL)

    PURPOSE:    Draw a class1 red crack, a class2 blue crack or a class3
                green crack.

****************************************************************************/
void DrawCrack(hWnd, x, y, crkclass, erase)
HWND  hWnd;
intx, y, crkclass;
BOOL  erase;
{
   HDC       hDC;
   COLORREF  crackcolor;
   int    i;

   if((x==0) && (y==0)) { return; };
   hDC = GetDC(hWnd);
```

```
    crackcolor = RGB(255,255,255);  /* Use White to erase*/
    if(crkclass == 1) {
       if(erase == FALSE){crackcolor = RGB(255,0,0);} /* red */
       else {crackcolor = RGB(255,255,255);};  /* white */
       for(i=-3; i<=3; i++) SetPixel(hDC, x+i, y+i, crackcolor);
       for(i=-3; i<=3; i++) SetPixel(hDC, x+i, y-i, crackcolor);
       for(i=-5; i<=5; i++) SetPixel(hDC, x, y+i, crackcolor);
       for(i=-5; i<=5; i++) SetPixel(hDC, x+i, y, crackcolor);
       }
    if(crkclass == 2) {
       if(erase == FALSE){crackcolor = RGB(0,0,255);} /* blue */
       else {crackcolor = RGB(255,255,255);};  /* white */
       for(i=-3; i<=3; i++) SetPixel(hDC, x+i, y+i, crackcolor);
       for(i=-3; i<=3; i++) SetPixel(hDC, x+i, y-i, crackcolor);
       for(i=-3; i<=3; i++) SetPixel(hDC, x, y+i, crackcolor);
       for(i=-3; i<=3; i++) SetPixel(hDC, x+i, y, crackcolor);
       }
    if(crkclass == 3) {
       if(erase == FALSE){crackcolor = RGB(0,100,0);} /* green */
       else {crackcolor = RGB(255,255,255);};  /* white */
       for(i=-3; i<=3; i++) SetPixel(hDC, x+i, y+i, crackcolor);
       for(i=-3; i<=3; i++) SetPixel(hDC, x+i, y-i, crackcolor);
       for(i=-2; i<=2; i++) SetPixel(hDC, x, y+i, crackcolor);
       for(i=-2; i<=2; i++) SetPixel(hDC, x+i, y, crackcolor);
       }
    ReleaseDC(hWnd, hDC);
}


/****************************************************************************

   FUNCTION:   DrawIndex(HWND, int, int, BOOL)

   PURPOSE:    Draw a black index on a selected crack.

****************************************************************************/
void DrawIndex(hWnd, x, y, erase)
HWND  hWnd;
intx, y;
BOOL  erase;
{
   HDC      hDC;
   COLORREF crackcolor;
   int   i;

   if((x==0) && (y==0)) { return; };
   hDC = GetDC(hWnd);

   if(erase == FALSE){crackcolor = RGB(0,0,0);}/* Black */
   else {crackcolor = RGB(255, 255, 255);};    /* White */
   SetPixel(hDC, x+5, y+5, crackcolor);
   SetPixel(hDC, x+4, y+4, crackcolor);
   SetPixel(hDC, x+5, y+4, crackcolor);
   SetPixel(hDC, x+4, y+5, crackcolor);
   SetPixel(hDC, x+5, y-5, crackcolor);
   SetPixel(hDC, x+4, y-4, crackcolor);
   SetPixel(hDC, x+5, y-4, crackcolor);
   SetPixel(hDC, x+4, y-5, crackcolor);
   SetPixel(hDC, x-5, y+5, crackcolor);
   SetPixel(hDC, x-4, y+4, crackcolor);
   SetPixel(hDC, x-5, y+4, crackcolor);
   SetPixel(hDC, x-4, y+5, crackcolor);
   SetPixel(hDC, x-5, y-5, crackcolor);
   SetPixel(hDC, x-4, y-4, crackcolor);
   SetPixel(hDC, x-5, y-4, crackcolor);
   SetPixel(hDC, x-4, y-5, crackcolor);
   for(i=-5; i<=5; i++) SetPixel(hDC, x+3, y+i, crackcolor);
   for(i=-5; i<=5; i++) SetPixel(hDC, x+i, y+3, crackcolor);
   for(i=-5; i<=5; i++) SetPixel(hDC, x-3, y+i, crackcolor);
   for(i=-5; i<=5; i++) SetPixel(hDC, x+i, y-3, crackcolor);
   ReleaseDC(hWnd, hDC);
}
```

```
/****************************************************************************

   FUNCTION:   DrawAllCrack(HWND)

   PURPOSE:    Draw all cracks and a index.

****************************************************************************/
void DrawAllCrack(hWnd)
{
   int i;
      for(i=1; i<=nTotal; i++){
            DrawCrack(hWnd, nX[i][0], nY[i][0], nClss[i], FALSE);
            DrawCrack(hWnd, nX[i][1], nY[i][1], nClss[i], FALSE);
            DrawCrack(hWnd, nX[i][2], nY[i][2], nClss[i], FALSE);
            };
      if(nSelect != 0){
            DrawIndex(hWnd, nX[nSelect][0], nY[nSelect][0], FALSE);
            DrawIndex(hWnd, nX[nSelect][1], nY[nSelect][1], FALSE);
            DrawIndex(hWnd, nX[nSelect][2], nY[nSelect][2], FALSE);
            };
}


/****************************************************************************

   FUNCTION:   EraseIndexSet(HWND, int)

   PURPOSE:    Erase a set of three indexes.

****************************************************************************/
void EraseIndexSet(hWnd, nSlct)
{

   DrawIndex(hWnd, nX[nSlct][0], nY[nSlct][0], TRUE);
   DrawIndex(hWnd, nX[nSlct][1], nY[nSlct][1], TRUE);
   DrawIndex(hWnd, nX[nSlct][2], nY[nSlct][2], TRUE);
}


/****************************************************************************

   FUNCTION:   EraseCrackSet(HWND, int)

   PURPOSE:    Erase a set of three cracks.

****************************************************************************/
void EraseCrackSet(hWnd, nSlct)
{
   DrawCrack(hWnd, nX[nSlct][0], nY[nSlct][0], nClss[nSlct],TRUE);
   DrawCrack(hWnd, nX[nSlct][1], nY[nSlct][1], nClss[nSlct],TRUE);
   DrawCrack(hWnd, nX[nSlct][2], nY[nSlct][2], nClss[nSlct],TRUE);
}

/****************************************************************************

   FUNCTION:   MainWndProc(HWND, unsigned, WORD, LONG)

   PURPOSE:    Processes window messages.

   MESSAGES:

       WM_COMMAND- Application menu item
       WM_DESTROY- Destroy window

****************************************************************************/
long FAR PASCAL MainWndProc(hWnd, message, wParam, lParam)
HWND        hWnd;
unsigned    message;
WORD        wParam;
LONG        lParam;
{
   FARPROC      lpProcAbout;
```

```
   PAINTSTRUCT Paint;
   HBITMAP      hButton;
   HDC          hMemoryDC;
   HANDLE   hDefault;

   static int         nView;
   int        i, j, mindist, nOldSelect;
   long          dist;
   BOOL          BullEye;

switch (message) {
   case WM_COMMAND:

      switch (wParam) {

/*** Menu File ***/

         case IDM_NEW:
            QuerySaveFile (hWnd);

            lpProcAbout = MakeProcInstance(SelectLayout, hInst);
            DialogBox(hInst, "SelectShipType", hWnd, lpProcAbout);
            FreeProcInstance(lpProcAbout);

            if (FileStuff.Name[0] != 0)
               { SetWindowText(hWnd,FileStuff.Name); }
            nSelect = 0;

            break;

         case IDM_OPEN:
            QuerySaveFile (hWnd);

            lpProcAbout = MakeProcInstance(OpenSource,hInst);
            DialogBox(hInst, "FileOpen", hWnd, lpProcAbout);
            FreeProcInstance(lpProcAbout);
            if (FileStuff.Name[0] != 0)
               { SetWindowText(hWnd,FileStuff.Name); }
            nSelect = nTotal;
            break;

         case IDM_SAVE:
            SaveFile(hWnd);
            break;

         case IDM_SAVEAS:
            lpProcAbout = MakeProcInstance(SaveAs,hInst);
            DialogBox(hInst, "FileSaveAs", hWnd, lpProcAbout);
            FreeProcInstance(lpProcAbout);
            if (FileStuff.Name[0] != 0)
               { SetWindowText(hWnd,FileStuff.Name);}
            break;

         case IDM_EXIT:
            QuerySaveFile (hWnd);
            DestroyWindow(hWnd);
            break;

/*** Menu Ship ***/

         case IDM_GENERAL:
            lpProcAbout = MakeProcInstance(ShipGeneral, hInst);
            DialogBox(hInst, "ShipGeneral", hWnd, lpProcAbout);
            FreeProcInstance(lpProcAbout);
            break;

/*** Menu Crack ***/

         case IDM_ADDCRACK1:
            nAddCrack = 1;
            nView = 1;
            SetCursor(LoadCursor(hInst, "VIEW1"));
            break;
```

```
            case IDM_ADDCRACK2:
               nAddCrack = 2;
               nView = 1;
               SetCursor(LoadCursor(hInst, "VIEW1"));
               break;

            case IDM_ADDCRACK3:
               nAddCrack = 3;
               nView = 1;
               SetCursor(LoadCursor(hInst, "VIEW1"));
               break;

            case IDM_DELETE:
               lpProcAbout = MakeProcInstance(DeleteCrk, hInst);
               DialogBox(hInst, "DeleteCrack", hWnd, lpProcAbout);
               FreeProcInstance(lpProcAbout);
               DrawAllCrack(hWnd);
               break;

            case IDM_EDIT:
               if(nAddCrack == 0)
               {
                  stpcpy(CrackType, szCrkType[nSelect]);
                  lpProcAbout = MakeProcInstance(EditRecord, hInst);
                  DialogBox(hInst, "CrackRecord", hWnd, lpProcAbout);
                  FreeProcInstance(lpProcAbout);
               }
               else
               {
               lpProcAbout = MakeProcInstance(AddRecord,hInst);
                  DialogBox(hInst, "CrackRecord", hWnd, lpProcAbout);
                  FreeProcInstance(lpProcAbout);

                  nAddCrack = 0;
                  SetCursor(LoadCursor(NULL, IDC_ARROW));

                  EraseIndexSet(hWnd, nSelect);
                  nSelect = nTotal;

                  DrawAllCrack(hWnd);
               };
               break;

/*** Menu Analysis ***/

            case IDM_Analysis:
               lpProcAbout = MakeProcInstance(Analysis,hInst);
               DialogBox(hInst, "RepairAnalysisDlg", hWnd, lpProcAbout);
               FreeProcInstance(lpProcAbout);
               break;

/*** Menu Help ***/

            case IDM_HELP_HOWTO:
               WinHelp(hWnd,"HELPHOW.HLP",HELP_INDEX,0L);
               break;

            case IDM_HELP_INFORMATION:
            WinHelp(hWnd,szHelpFileName,HELP_INDEX,0L);
            break;

            case IDM_ABOUT:
               lpProcAbout = MakeProcInstance(About, hInst);
               DialogBox(hInst, "AboutBox", hWnd, lpProcAbout);
            FreeProcInstance(lpProcAbout);
            break;

            default:
            return (DefWindowProc(hWnd, message, wParam, lParam));
            }
         EnableGrayed(hWnd);
         break;
```

```
case WM_LBUTTONDOWN:
   return (DefWindowProc(hWnd, message, wParam, lParam));

case WM_LBUTTONUP:
   if((nAddCrack > 0)&&(nView == 1)){
      nTotal ++;
      nX[nTotal][0] = LOWORD(lParam);
      nY[nTotal][0] = HIWORD(lParam);
      nClss[nTotal] = nAddCrack;
      nView = 2;
      DrawCrack(hWnd, nX[nTotal][0], nY[nTotal][0], nClss[nTotal], FALSE);
      SetCursor(LoadCursor(hInst, "VIEW2"));
      }

   else if((nAddCrack > 0)&&(nView == 2)){
     nX[nTotal][1] = LOWORD(lParam);
     nY[nTotal][1] = HIWORD(lParam);
     nView = 3;
     DrawCrack(hWnd, nX[nTotal][1], nY[nTotal][1], nClss[nTotal], FALSE);
     SetCursor(LoadCursor(hInst, "VIEW3"));
     }

   else if((nAddCrack > 0)&&(nView == 3)){
     nX[nTotal][2] = LOWORD(lParam);
     nY[nTotal][2] = HIWORD(lParam);
     DrawCrack(hWnd, nX[nTotal][2], nY[nTotal][2], nClss[nTotal], FALSE);

     /* Go to function 'AddRecord()' */
     lpProcAbout = MakeProcInstance(AddRecord,hInst);
     DialogBox(hInst, "CrackRecord", hWnd, lpProcAbout);
     FreeProcInstance(lpProcAbout);

     nAddCrack = 0;
     SetCursor(LoadCursor(NULL, IDC_ARROW));

     EraseIndexSet(hWnd, nSelect);
     nSelect = nTotal;

     DrawAllCrack(hWnd);
     }

  else if(nTotal > 0){
    nOldSelect = nSelect;
    mindist = 40;

    for(i=1; i<=nTotal; i++)
    {
       for(j=0; j<=2; j++)
       {
       dist =    (nX[i][j] - LOWORD(lParam)) *
                 (nX[i][j] - LOWORD(lParam)) +
                 (nY[i][j] - HIWORD(lParam)) *
                 (nY[i][j] - HIWORD(lParam));
       if(dist < mindist) {
          mindist = dist;
          nSelect = i;
          BullEye = TRUE;
          };
       }
    }
    if(BullEye == TRUE){
       EraseIndexSet(hWnd, nOldSelect);
       DrawAllCrack(hWnd);
       };
    if((nOldSelect==nSelect)&&(BullEye==TRUE)){
       stpcpy(CrackType, szCrkType[nSelect]);
       lpProcAbout = MakeProcInstance(EditRecord,hInst);
       DialogBox(hInst, "CrackRecord", hWnd, lpProcAbout);
       FreeProcInstance(lpProcAbout);
    }
   }
break;
```

```
    case WM_NCLBUTTONDOWN:
       return (DefWindowProc(hWnd, message, wParam, lParam));

    case WM_KEYDOWN:
       break;

    case WM_SETCURSOR:
       return (DefWindowProc(hWnd, message, wParam, lParam));
       break;

    case WM_PAINT:

       if(userBitmap == 1)    userPaint(hWnd, ShipType);
       else                   myPaint(hWnd, ShipType);

       DrawAllCrack(hWnd);

       return (TRUE);

    case WM_INITMENU:
       if (bHelp) { SetCursor(hHelpCursor); }
       return (TRUE);

    case WM_ENTERIDLE:
       if ((wParam == MSGF_MENU) && (GetKeyState(VK_F1) & 0x8000)) {
       bHelp = TRUE;
          PostMessage(hWnd, WM_KEYDOWN, VK_RETURN, 0L);
       }
       break;

    case WM_DESTROY:
       QuerySaveFile (hWnd);
       WinHelp(hWnd,szHelpFileName,HELP_QUIT,0L);
       PostQuitMessage(0);
       break;

    default:
       return (DefWindowProc(hWnd, message, wParam, lParam));
    }
    return (NULL);
}
```


190

Repair Management System for Critical Structural Details in Ships

Final Report

Kai-tung Ma
and
Professor Robert G. Bea

Department of Naval Architecture and Offshore Engineering
University of California, Berkeley

191
192X

PREFACE

The Joint Industry Research Project "**Repair Management System for Fatigue Cracks in Ship Critical Structural Details (RMS 3.0)** was conducted from December 1993 to December 1994. The objective of this project was to develop practical tools and procedures for the analysis of proposed ship structural repairs and to prepare guidelines for the cost-effective repair and design. It was carried out in the Department of Naval Architecture and Offshore Engineer (NAOE), University of California at Berkeley. Graduate Student Researcher, Kai-tung Ma, performed the work together with the supervision and help of the principal investigator, Professor Robert G. Bea

The source of the RMS projects can be traced back to a two-year Joint Industry Research Project "Structural Maintenance Project for New and Existing Ships (SMP-I)". The SMP-I was initiated in 1990 by the Department of NAOE. It included six related studies. The fourth study, Fatigue and Corrosion Repair Assessments, created the RMS 1.0. During the next one-year project, SMP-II, version 2.0 of RMS was developed. RMS 3.0 was sponsored by the following five organizations:

- **American Bureau of Shipping**
- **Lisnave - Estaleiros Navais de Lisboa, SA**
- **Ship Structure Committee**
- **Lloyd's Register of Shipping**
- **Newport News Shipbuilding & Dry Dock Co.**

This report is the final report of the RMS 3.0. It documents the development of RMS 3.0 and summarizes the other two previous RMS projects.

TABLE OF CONTENTS

Page

LIST OF FIGURES

Page

CHAPTER 1. INTRODUCTION

This chapter discusses current problems associated with repairs to Critical Structural Details (CSD) in ships. It introduces the objectives of a Repair Management System (RMS). The components of RMS are defined. The three major accomplishments of this project are summarized. Each one of the chapters of this report are briefly reviewed.

1.1 Problem Definition

With the introduction of larger steel ships like very large crude carriers (VLCC), the tasks of maintaining these ships have become increasingly difficult. Many of these ships have experienced varying degrees of fatigue cracking problems. Due to the limited time available in dry-docks, lengthy fatigue analyses have been rarely used. Repair decisions on those fatigue cracks are often based on experience-based knowledge and lack analytical evaluation.

To minimize the risk of future structural failures due to poor repair, an advanced approach of repair analysis was developed during the RMS projects to help repair engineers evaluate repair alternatives. It was the goal of this project to review the process of structural maintenance and repair of ship structures and to develop a new approach to help manage the information used to make good decisions. Specifically, a practical tool to evaluate fatigue crack repair strategies has been developed to improve the durability of existing ships.

Recently, considerable effort has been put into understanding the effectiveness of specific repairs, especially those associated with fatigue of CSDs. This effort has resulted both from an aging fleet of existing ships and a heightened public interest in environmental issues [USCG, 1990][Jordon, 1978, 1980][TSCF, 1991]. In addition, records of ship condition are shifting from paper-based systems to computerized systems that contain inspection and repair information in database format [Schulte-Strathaus and Bea, 1994]. The information about ship maintenance and repair can be sorted by an experienced repair engineer to help evaluate the effectiveness of past repairs and assess the overall condition of the ship.

The most technical part of the ship maintenance and repair is the decision making associated with evaluating and choosing a suitable and reliable repair method for a particular CSD failure. Ship structural repair decisions are difficult due to the vast array of engineering knowledge which must be assimilated in order to make a good repair decision. This knowledge includes (1) experience-based knowledge about repairs and ship condition, (2) a large volume of historical information from past ship inspections and repairs, (3) complex ship structure information, (4) complex loading information, (5) complex analysis procedures, and (6) expert knowledge of structural design, fracture mechanics and corrosion.

Poor or incomplete repair decisions are often made simply because there is not enough time or money to perform a detailed analysis. This indicates that a tool needs to be developed for the development and management of the information used to make intelligent repair decisions.

1.2 Repair Management System

This poses the key question addressed during this project: 'How do we properly manage the computerized inspection and repair data, the existing knowledge of both successful and unsuccessful repairs, the complex analysis tools and additional knowledge to make intelligent and timely repair decisions?"

The answer proposed by this project is the **Repair Management System (RMS)**. The RMS is a computerized framework to help repair engineers make good repair decisions by providing them with structural failure diagnosis and repair alternative evaluation. The RMS is the first known attempt to handle the complexities of ship structural repair analysis in a framework that provides both elements critical to good repair--quick decisions and thorough evaluations.

The RMS is a system that it has several modules (Figure 1.1). Each module performs a certain task but cannot reach a specific goal. However, all modules together form a system which can help repair engineers process a variety of repair alternatives, estimate the fatigue lives of each repair, and make an optimal decision.

Users start the system with a repair alternative module (see Chapter 2) that provides available repairs in a graphic mode. Then, the core module which is a fatigue calculation module (see Chapter 3) will try to develops an estimate of the fatigue life for each repair. This module needs input from an S-N curve selection module (see Chapter 4) and a stress reduction factor module (see Chapter 5). Finally a decision aiding module (see Chapter 2) and a reliability module (see Chapter 3) help users make an optimal decision.

To limit the scope of RMS, concentration has been placed on some ship CSDs with high failure rates. To further define the scope, a questionnaire was sent to all the participants in the early stage of this study [Gallion, 1992]. This questionnaire requested information on the most desirable features of computer software associated with repairs. The result of the ranking of the proposed features is listed as follows (#1 most favorable; #6 least favorable):

1. Expected life analysis of repair alternatives
2. Graphical database of possible repairs

3. Economic tradeoff analysis of repair alternatives
4. Reliability-based information
5. Extendibility to allow updating with new repair data
6. Repair database analysis capabilities (statistical)

The highest priorities of participants that responded were the expected life analysis of repairs. As a result, this project was focused on the development of the first feature within the RMS. The second, third and fourth features were explored. The last two features were not addressed.

1.3 RMS 3.0 Major Developments

Through the course of exploring the selected features, three major developments were accomplished during this project. The first achievement is development of a faster way to calculate fatigue life of repaired CSDs. The new method eliminates the computation of the long-term spectral load involved in a traditional fatigue analysis. It allows much faster fatigue-life estimation for repaired CSDs. As a result, it can be used to make quick repair decisions in repair yards and will save the valuable service time of the ship.

The second achievement is development of a new method to calibrate 'Cyclic Stress Range - Number of Cycles to Failure (S-N)' information. The selection of proper S-N curves for a particular CSD is a common difficulty in fatigue analyses. Traditionally, a certain class of S-N curve is compared and matched to a critical location of a CSD with similar geometry as fatigue specimens. Inaccuracy may be introduced in this matching process. The new method considers the CSDs in in-service ships as fatigue specimens. With gathered historical repair data, the S-N curves of particular repaired

CSD can be determined. Two sets of repair data were gathered and analyzed. The results showed that the rewelding of flatbar-welds has an S-N curve lying between F2 and G classes. The veeing and welding of a cutout radius has an S-N curve lying between E and F classes.

The third achievement are results from a series of Finite Element Analyses (FEA) on three CSDs. For each CSD, eight variety models were built by interchanging a lug, a flatbar and a backing bracket. A total of 24 models were built. A Stress Reduction Factor (SRF) was defined as the ratio of hot spot stresses before and after adding a structural component such as a lug. Using the results of the FEAs, SRFs for typical repairs were determined.

With these developments, the RMS is able to provide engineers with a computerized tool for analyzing repair alternatives. The functions and advantages of the RMS are: (1) providing a consistent repair strategy, (2) ensuring more complete evaluations in a timely manner, (3) increasing the level of expertise in the shipyard and office, (4) promoting sharing of repair information among ship owners, operators and shipyards, (5) utilizing historical ship data, and (6) utilizing both numeric and symbolic information.

1.4 Contents of Report

In Chapter 2 the basics of ship structural inspection and maintenance are discussed. These basics include an introduction of inspection and crack repair. The general strategies for crack repair are outlined. The examples of the repair alternative selection are shown on some critical structural details. The specifics of CSD repair are discussed. In addition, repair decision making is discussed. Basic steps to determining the best repair are explained.

In Chapter 3 a new method of simplified comparative fatigue analysis is developed to estimate the fatigue lives of the repair alternatives. The method is based on Wirsching's fatigue equation [Wirsching, 1983]. It successfully eliminates the need of lengthy load computation involved in a full spectral fatigue analysis. The load parameters (extreme stress and the Weibull parameters) can be obtained from the first failure life by reversing the procedure of a fatigue analysis. The procedures of computing repair life are outlined. Also an example of repair alternative evaluation is reviewed. A reliability model is reviewed, since significant uncertainty exists in the factors of fatigue damage expressions. Uncertainties of variables in the Wirsching's fatigue expression are treated by considering each as a lognormal distributed random variable. A concise reliability expression is resulted. Adapting this reliability model to the RMS is considered.

In Chapter 4 procedures to define S-N curves for ship CSDs are discussed. The S-N designation developed by American Bureau of Shipping (ABS) is reviewed [ABS, 1993]. The ABS designation is for newly constructed CSDs. However in order to estimate repair lives, the RMS requires a S-N designation for repaired CSDs (by vee and weld). A new approach that utilizes historical inspection data to establish the relation between S-N curves and CSDs is introduced. Gathering and analyzing of two sets of inspection sample data are summarized.

In Chapter 5 a stress reduction factor (SRF) is defined as the ratio between the stresses after and before repairs by design modification. The SRF is a crucial input parameter for the fatigue life estimation. In order to obtain the SRFs for common repairs done on some typical CSDs, three CSDs were selected and analyzed. For each CSD, eight finite element models were built. The models were analyzed and the resulting stresses were compared to get the SRFs. A table is developed to summarize these SRFs.

In Chapter 6 the RMS approach is used in the development of a computer program to illustrate the evaluation of repair alternatives for fatigue failure of some CSDs. A case study analysis is conducted to verify the code and illustrate its effectiveness as a repair tool. The RMS crack repair database for the fatigue mode of structural failure is outlined.

Finally, in Chapter 7 the project is summarized with some concluding remarks and recommendations for future developments.

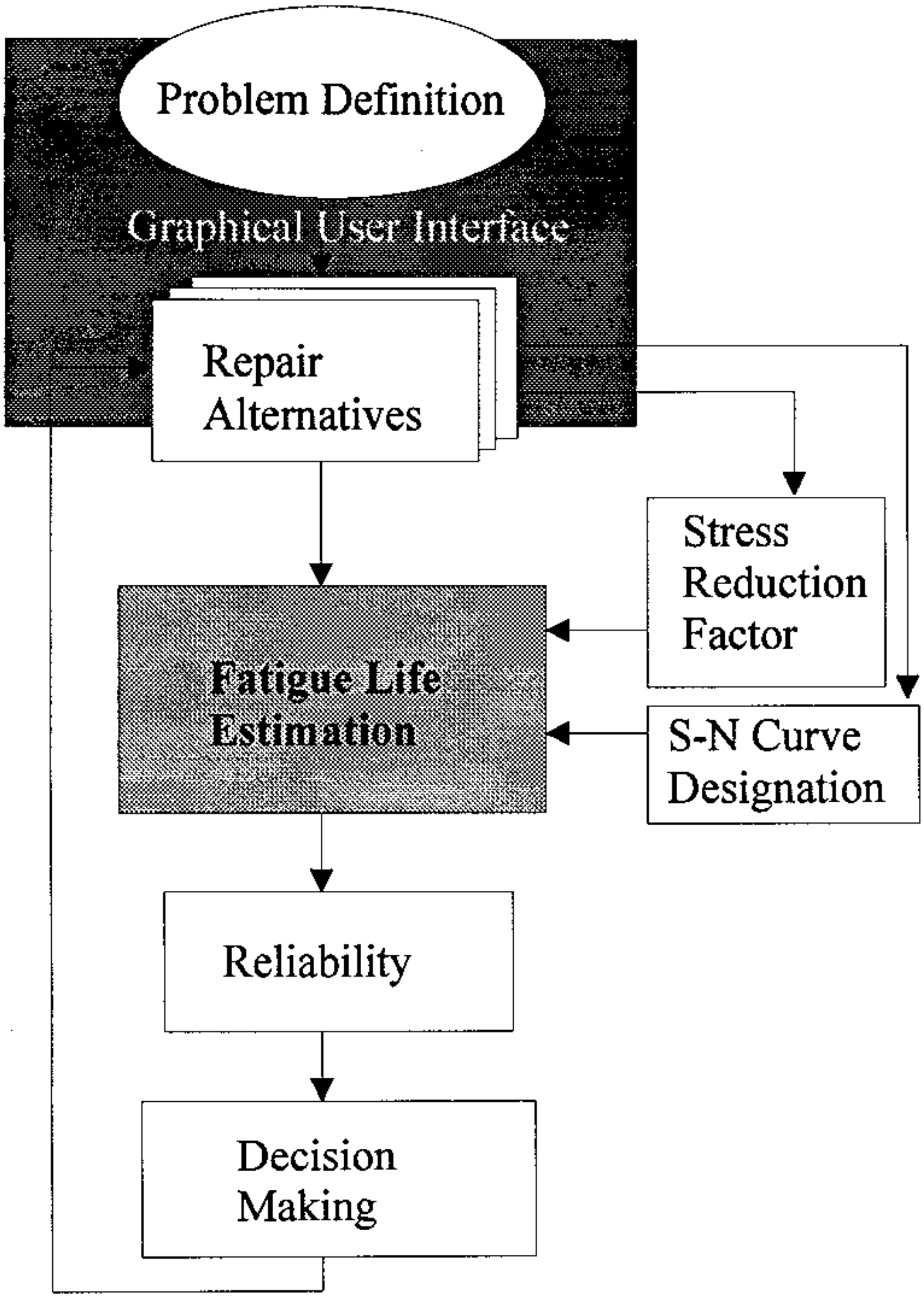


Figure 1.1: RMS System Architecture

CHAPTER 2. REPAIR ALTERNATIVES AND DECISION MAKING

The purpose of this chapter is to review the inspection, maintenance and repair of CSDs in ships. It introduces typical repair alternatives in current practice. It also looks at the factors that go into development of an intelligent repair decision.

2.1 Fatigue in Ships

Fatigue cracks are potentially the most serious of defects because they can grow rapidly in size leaving the affected structure unable to bear loads. As the result of a fatigue crack, the structure around a crack must carry a greater loading that can in turn lead to its failure in the future. If this cracking process continues unchecked, hull girder or large panels of side shell collapse can result. Therefore ship's hull structure has to be inspected periodically and repaired when cracks are found.

Ship's hull structure must be monitored by a series of internal and external inspections to assess the integrity of the ship structure. The scope and frequency of these inspections are regulated by classification society or U.S. Coast Guard. In addition to those, ship owners may have their voluntary inspection due to un-foreseen emergency or owner's plans. All these inspections provide means to evaluate the current condition of steel and coatings and to detect unexpected cracks and damages.

During an inspection, several types of structural failures can be found. Fatigue cracks, corrosion, coating breakdowns and buckling are the most common failures. Fatigue failure of CSDs is the one that is the objective of this study. In some cases, hundreds of cracks can be found in an old tanker. A majority of these cracks are due to keep in mind the other possible modes of cracking. The mode of failure dictates the analysis procedures required to evaluate a failure.

2.2 Repair Alternatives

According to the regulations and practices of the U.S. Coast Guard, it is required that most detected cracks have to be repaired before leaving port. Repair of cracks vary widely. Repairs of cracks can range from temporary drilling a stopping-hole in front of crack tip to complete re-design of the structural detail and replacement of steel near the detail.

Experience indicates that many repairs fail again and repairs must be repeated. In one case, a series of side shell longitudinal cracks has been repaired four times, and each time a different repair procedure has been tried [Bea, 1992].

For a fatigue crack in a particular CSD, there are several methods to repair it. The expected life and cost of each repair method varies. The general strategies for crack repair of critical structural details can be classified in the following way:

- **Drill a stopping hole in front of the crack tip (Temporary repair):** Cracks may be arrested temporarily by drilling a hole of diameter equal to the plate thickness at a distance of two plate thickness in front of the visible crack tip and on a line with the direction of anticipated crack propagation. Such repairs are sometimes considered in attempting to get the ship to a facility where full repairs can be made. It may also be used for cracks in secondary structure (the structure which neither contributes to the strength nor the watertight integrity such as partition bulkheads, platforms and so on).

- **Re-weld the cracks to the original construction:** Gouging with re-welding (veeing and welding) is an easy and common way of repair. However, the strength of re-welding cracks is, almost invariably, worse than the original one. The repaired weld will create new crack potentials and thus fail again in a shorter time interval.
- **Re-weld the cracks plus post weld improvement:** This repair is basically the same as the pervious one, except that the weld is ground smooth to improve its fatigue strength. The life extension effect of post weld improvement can be significant [Almar-Naess, 1985].
- **Replace the cracking plate:** This is also called inserting a new plate. The inserted new plate has a new fatigue life which is the same as the original life. Since this plate has never carried any loads, its fatigue damage factor is zero. If the loading history and the material is identical to those of the failed plate, its fatigue life should be about the same as the failed time of the crack.
- **Modify design by adding bracket, stiffener, lug, or collar plate:** The more robust way of repair is to modify the local geometry to reduce the stress concentration. While adding a detail component and not involving cropping a large section, this repair may be one of the best. It can reduce the stress concentration and therefore increase the repair life significantly. In addition it is reasonably easy to apply.
- **Change configuration by applying soft toe, increasing radius, trimming face plate, enlarging drain holes, etc.:** This is another way to modify the local geometry to reduce the stress concentration. If a longer life continuance is expected for the ship, a more robust repair such as this should be considered.
- **Enhance scantling in size or thickness:** Increasing the size of a detail like a bracket is good. However increasing the thickness may not be a very good repair in

the situation where a discontinuity is introduced to the plate. While doing this, the discontinuity should be carefully located outside the high stress area.

Two examples of repair alternatives are shown in Figures 2.2 and 2.3. The first example is a crack in a longitudinal cutout. This type of crack comprises 12.3% of total cracks in ships according to prior studies [Jordon, 1978 and 1980]. The second one is a crack on a longitudinal near the tip of a beam bracket. This type comprises 32.8% of total cracks.

It is difficult to decide which repair method is most reliable or cost effective for a particular failed CSD. Therefore, a way to determine the optimal repair needs to be developed.

2.3 Repair Decision Making

When a structural failure in the form of cracking is discovered by inspection, a decision must be made as to the most effective repair. This decision is difficult due to the vast array of engineering, construction, and repair considerations. However, many additional factors must also be considered in a very short time. These factors include technical, economic, schedule and logistic factors.

As a result of the complexity and the short time allowed, ship repairs currently rely heavily on the experience of repair engineers and repair yard personnel. This is the experience-based approach. There is simply not enough time to take into account all possible factors and perform detailed analyses. Repair decisions often lack in technical and economic evaluation, but serve to get ships back into service quickly.

In contrast to the experience-based approach, it is possible to perform a full fatigue analysis to determine the expected repair lives of different repair alternatives. The full fatigue analysis approach involves detailed ship motion analysis, global and local fatigue. Therefore, fatigue failure has been a serious maintenance problem for some ship owners.

Cracks can be initiated in several modes. Fatigue fracture is the most common mode in ships. It is important to keep in mind that there are other possible fracture modes for cracks. Brittle fracture occurs under static loading and is typical in materials with yield strengths less than 0.5 percent strain before fracture, such as cast iron, concrete and ceramic. Materials that are not normally brittle can become brittle in some environments, such as low temperatures. The fracture surface is usually flat and contains arrow shaped lines known as "Chevron marks" which point to the origin of the failure. Ductile fracture is another possible mode. It occurs under static loading and is typical in materials with yield strengths greater than 0.5 percent strain before fracture, such as steel and aluminum. Failure is predicted by several theories, including the maximum shear stress theory and the distortion energy theory (von Mises). The fracture surface is usually distorted by yielding.

All the above modes can be accelerated by corrosion. General corrosion reduces plate thickness and increases stresses on the plate. In addition, a single fracture can contain several modes. For example, a small crack that exists at a welding imperfection will grow in a stable manner by fatigue. At some crack length, the stress may reach a critical level and cause unstable crack growth by brittle fracture. This brittle fracture may be arrested by load sharing with adjacent structure or an increase in material thickness along the crack front.

Since past experience indicates about 70% of the total damage in ships over 200 meters in length may be classified as fatigue damage [Committee III.1, 1979], this project addressed only fatigue cracks. This report concentrates on the proper repair of fatigue cracks in critical structural details (CSDs) of ships (Figure 2.1). It is important to finite element models, and fatigue analysis [ABS, 1993]. It usually takes a few months, and will cause ship owners a significant economic loss due to out of service time of the ships. Therefore, this approach is rarely used.

Clearly, the traditional approach lacks adequate technical justification and the detailed approach, although necessary at times, is inadequate to make on-the-spot repair decisions. The goal of RMS is to provide a computerized system to allow a more complete evaluation of the repair alternatives in a short time period. In order to accomplish this goal, the approach taken by RMS must provide a quick and sufficiently reliable method to estimate fatigue life without going through detailed ship motion analysis and global finite element analysis. This method will be described in the next chapter.

Given a quick and reliable fatigue life estimation method, repair alternatives can be ranked according to the expected life and cost of the repair. The user must select the most appropriate alternative from his or her knowledge of the economics of the ship. For example, for a fracture which took ten years to develop and discover, the repair options might be:

(1) Grind out crack and re-weld--5 years expected life

(2) Cut out section and butt-weld new piece--10 years expected life

(3) Add one bracket --12 years expected life

(4) Add two brackets --15 years expected life

Depending on the economic goals of the owner, a different repair alternative will be selected. For example, if the ship has only two more years in service, the cheapest alternative with an expected life of greater than two years will be selected.

The above case illustrates the simplest optimal cost model for making repair decisions. The general case of this optimal cost model is shown schematically in Figure minimize the chance that such an incident is caused by poor repair and maintenance of the structure. Accessibility for monitoring by crew will determine whether monitoring of minor structural problems is feasible. If a structural failure cannot be monitored effectively it must be repaired.

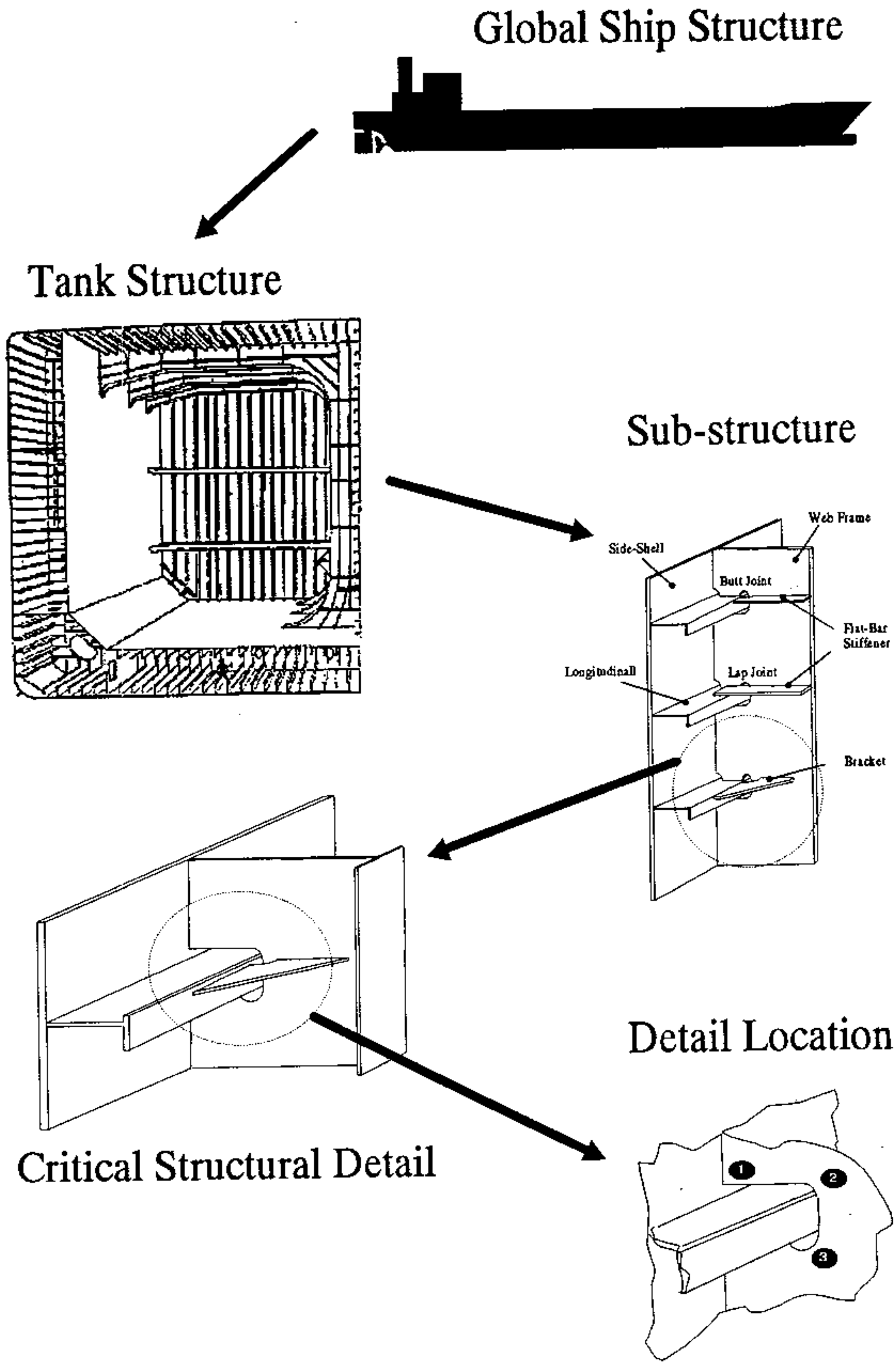


Figure 2.1: Global Structure to Side Shell Structure Components

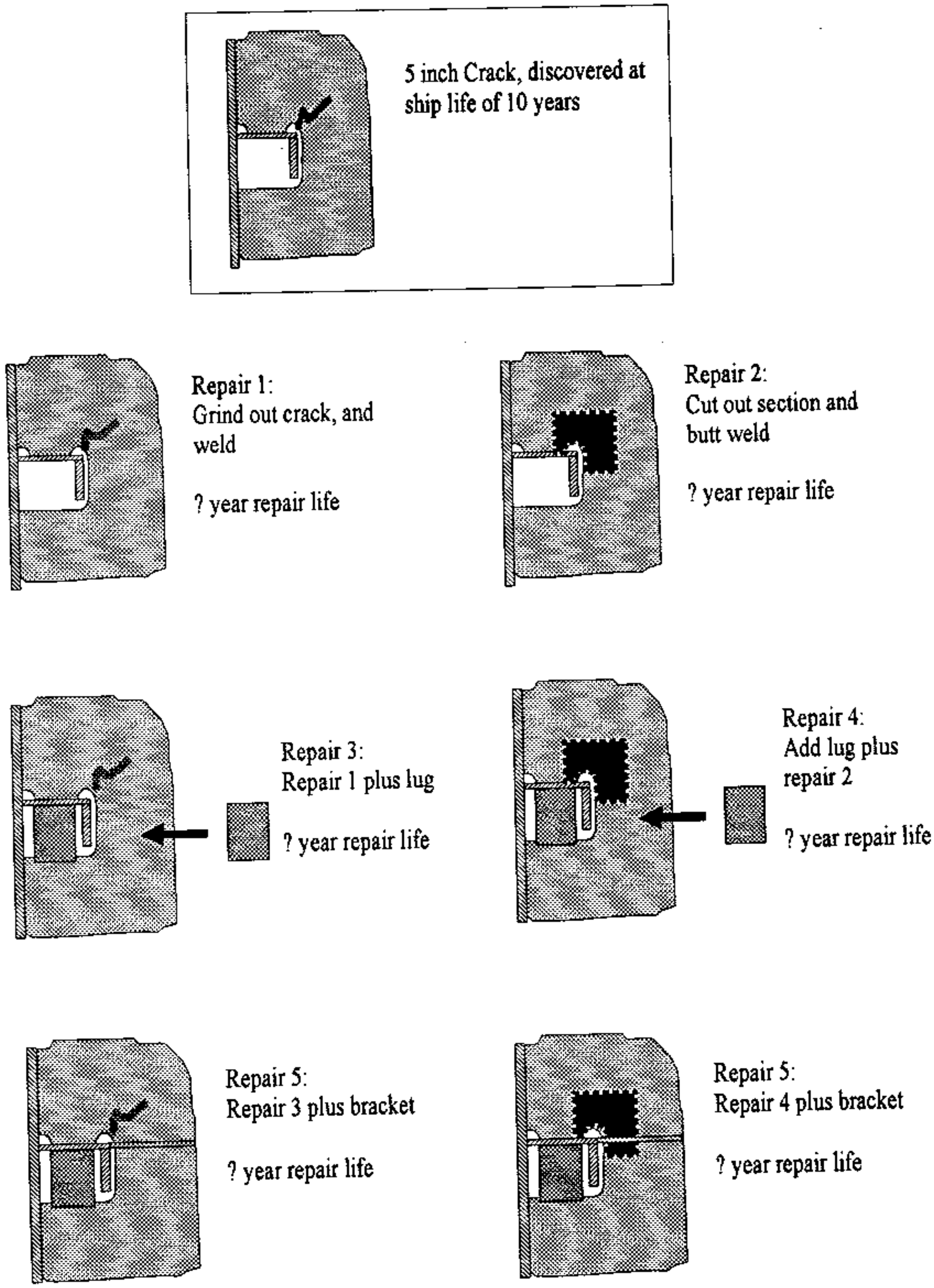


Figure 2.2: Repair alternatives example 1 [Ma, 1992]

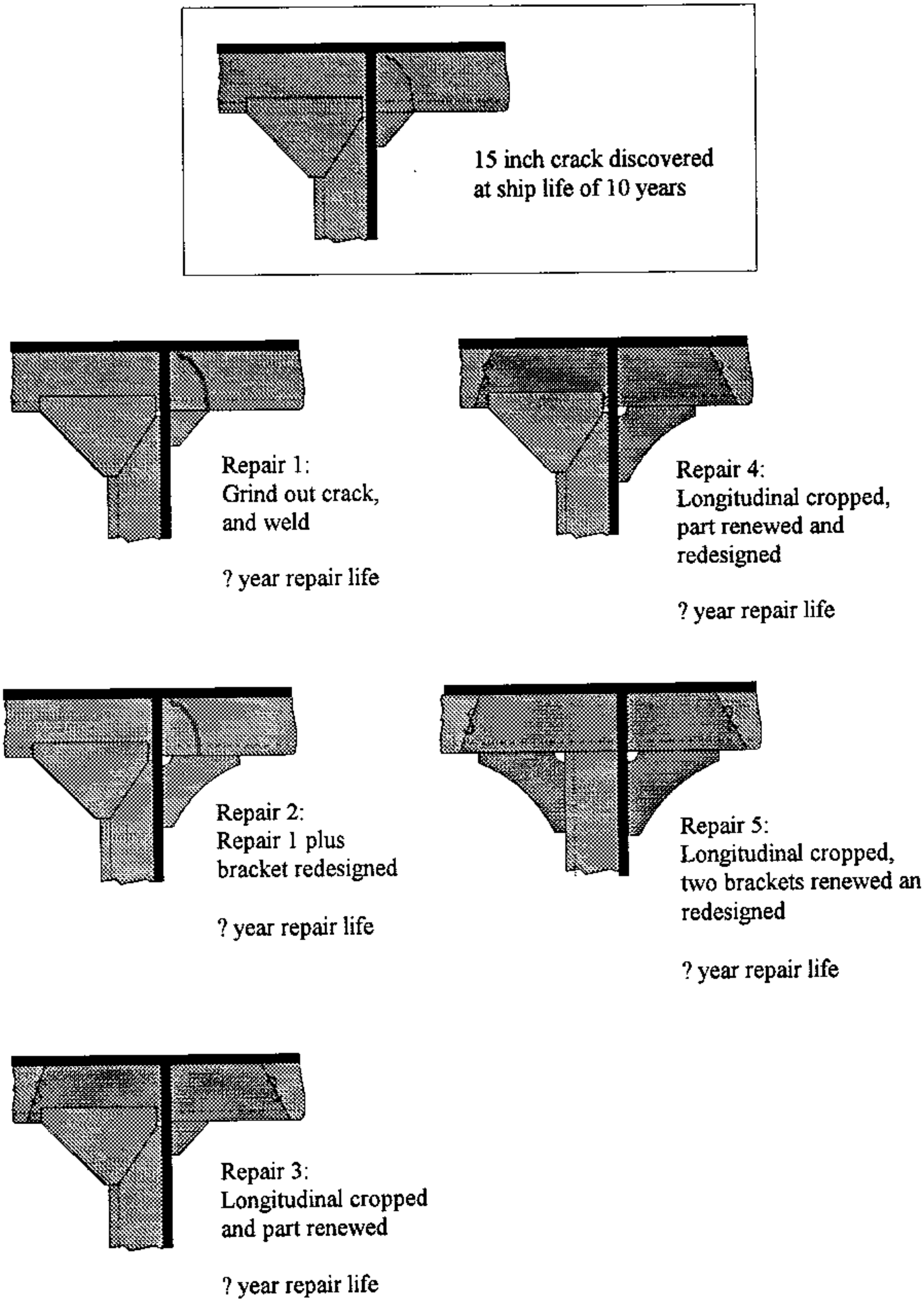


Figure 2.3: Repair alternatives example 2

2.4. The best repair option is defined as the one that results in the minimum total costs (initial plus future) over the life of the ship.

Take the repair of a crack in a longitudinal clearance cutout for another example. Assume a ship is going to be operated for 10 more years, and the owner chooses the cheapest and easiest repair like veeing and welding. The initial repair cost will be very low. Since this repair is not robust, its expected repair life will be short. Then the owner may have to repair it again and again if continuing the same repair method. The future cost is therefore quite high. This is the option of the non-durable repair in Figure 2.4. On the other hand, if the owner chooses a very robust repair like inserting plate and adding a lug, the initial cost may be high. But the future repair cost will be as low as zero in this case. This becomes the case of the durable repair. The 'best" repair is the one that produces a minimum total cost.

The concept of optimum repair based on the lowest total repair cost is quite simple. There can be many other factors that should be taken into consideration in the real world. The repair that best satisfies the life continuance, economic, location, time and other considerations is the one that should be chosen. These repair considerations are discussed in the following paragraphs [Gallion, 1992].

Life continuance consideration can be the most important factor in repair decisions. For example if a ship is going to be kept in service for another 5 years and then to be retired or sold, the ship owner may select a repair that can last for more than 5 years. Supposing the repair work well, the failed critical structural detail will be out of trouble for the rest of 5 years with a high reliability. This consideration is related to the economic consideration. However, the difficult part is the life estimation of a particular repair method. This will be discussed later in this report.

Economic considerations can play a dominant role in repair decisions. These economic factors include the future plans for the ship, age of the ship, total cost and time to complete repairs, cargo transport obligations, money available, current steel costs, repair rates, wage rates, etc. The economic decision is usually based on the certain initial repair costs and not the possible future costs of maintenance. This is mainly due to the complexity of the repair decision, which makes future costs difficult to evaluate. However, future costs for inadequate, non-durable repairs may dominate the decision. A complete economic analysis should take into account the tradeoff between initial and future costs. In the same way that a more durable ship has lower maintenance costs, more durable repairs will have lower future repair costs.

Repair location consideration falls into two categories. Voyage repairs are made at sea mostly in emergency situations. Voyage repairs are often very difficult since "hot work" (welding) is usually prohibited in critical hull structures due to the presence of flammable materials. As a result, cold patching is a popular temporary remedy. Shipyard repairs are made either at dockside or in a dry-dock environment after the tanks are ventilated and washed to accommodate hot work in the tanks. This is the most ideal repair environment although it still presents problems due to the enormous size of crude oil carriers.

Time considerations include factors such as the time available to complete repairs and the time until the next inspection and repairs. More thorough repairs are required if there is a long time before the next inspection or overhaul period.

Several additional considerations must be taken into account in repair alternative evaluations. These considerations include the following: Classification societies and regulating authorities dictate the minimum structural requirements for compliance with class rules. Environmental safety has become a major consideration in the repair of ships. Environmental disasters can produce both ecological damage and serious financial damage to the owner and operators of the ship as illustrated by the grounding of the Exxon Valdez in Prince William Sound. The goal of repairs is to

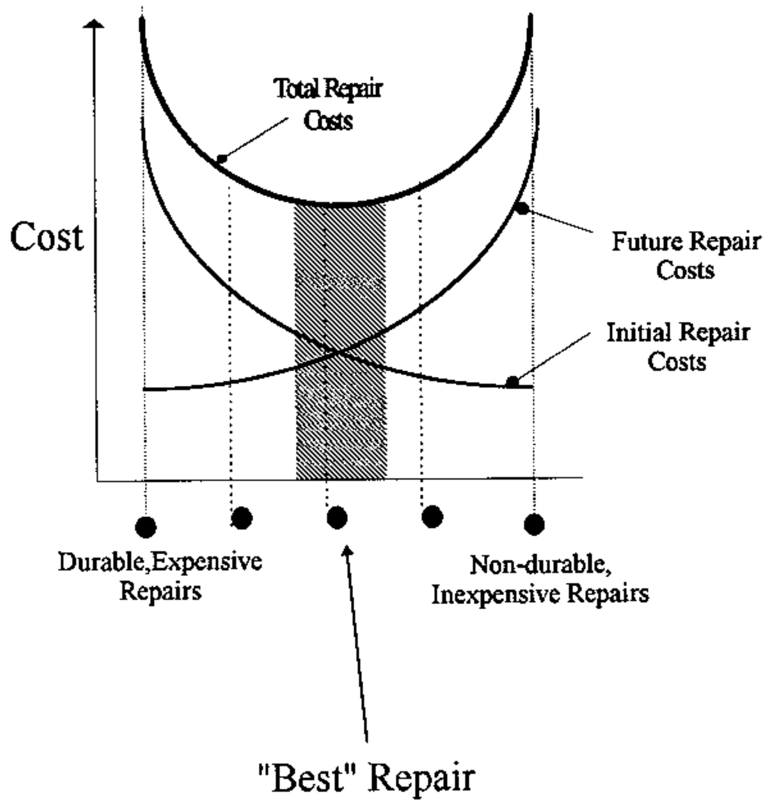


Figure 2.4: Repair Cost Tradeoff

221

CHAPTER 3. REPAIR LIFE ESTIMATION

The key to any repair analysis is the ability to rank repair alternatives according to some index. In the Repair Management System (RMS), the expected life of a repair is used as the index.

In this chapter we will develop an approach of fatigue life estimation based a simplified fatigue analysis. The simplified fatigue analysis will be enhanced and modified so that it takes a much shorter time than usual to perform. Ihis is achieved by eliminating lengthy load computations. As a result, quick comparison of repair alternatives is possible. This approach can be applied to practical uses and can promote the expertise in repair yard. An example of repair analysis is reviewed. Lastly, a fatigue reliability model is summarized.

3.1 Fatigue Calculation

To calculate the fatigue life (or damage) of a ship CSD, the stress history needs to be obtained or modeled. This stress history can be denoted by a sequence of stress ranges, S_i (where i is from 1 to its total number of cycles, N_T). Miner's rule states that the damage due to the i^{th} cycle is:

$$D_i = \frac{1}{N(S_i)} = \frac{S_i^{\,m}}{A} \tag{3.1}$$

$N(S_i)$ = Number of cycles alternating stress S_i applied

A = Life intercept, mean

The total damage is obtained by summing the damage of each stress cycle:

$$D = \frac{1}{A}\sum_{i=1}^{N_T} S_i^{\,m} \tag{3.2}$$

For large N_T, the following approximation can be made:

$$E\left[S^m\right] \cong \frac{1}{N_T}\sum_{i=1}^{N_T} S_i^{\,m} \tag{3.3}$$

$E[S^m]$ is the expected value of S^m, and S is now defined as a random variable denoting fatigue stress range. Assuming equality, Equation 3.3 becomes:

$$D = \frac{N_T E\left[S^m\right]}{A} \tag{3.4}$$

Assume that the S has a Weibull distribution. The expected value of S^m is:

$$E\left[S^m\right] = \lambda(m)\delta^m \Gamma\left(\frac{m}{\varepsilon}+1\right) \tag{3.5}$$

Where δ = scale parameter,

ε = Weibull (stress range) parameter, and

$\Gamma(\,)$ = Gamma function.

The rainflow correction factor $\lambda(m)$ is one, if δ and ε are evaluated directly from measured or estimated stress histories. The parameter δ can be expressed in terms of S_0 (once in a lifetime stress or extreme stress). S_0 is the alternating stress that is exceeded on an average of once N_0 cycles.

$$S_0 = (\ln N_T)^{1/\varepsilon}\,\delta \tag{3.6}$$

The average frequency of the stress cycles is defined as $f_0 = N_T/T$. The average frequency f_0 of the stress cycles is a constant $2.5\text{x}10^6$ cycles/yr for the wave loading on ship structure.

A random variable, B, is introduced to account the uncertainties associated with the computation of fatigue stresses. Making the substitutions and combining Equation 3.5 and 3.6, Equation 3.4 becomes [Wirsching, 1983]:

$$D = \frac{TB^m}{A} f_o \, S_o^m \, [\ln N_T]^{-(m/\varepsilon)} \, \Gamma\left(\frac{m}{\varepsilon} + 1\right) \tag{3.7}$$

where T = Time to failure

B = Uncertainty factor in estimation of fatigue stress

Equation 3.7 has been further developed to account for the effect of two-slope S-N curves. In that case the above equation will have an extra term μ:

$$D = \frac{TB^m}{A} f_o \, S_o^m \, [\ln N_T]^{-(m/\varepsilon)} \mu \, \Gamma\left(\frac{m}{\varepsilon} + 1\right) \tag{3.8}$$

where $\mu = 1 - \{\gamma(1 + m / \varepsilon, \upsilon) - \upsilon^{-\Delta m/\varepsilon} \gamma(1 + [m + \Delta m] / \varepsilon, \upsilon)\} / \Gamma(1 + m / \varepsilon)$

$\upsilon = \left(S_q / S_0\right)^{\varepsilon} \ln N_T$

S_q = S-N stress range at the intersection of two segments,

Δm = Slope change of the upper to lower segment of the S-N curve, and

$\gamma(a, x)$ = In-complete gamma function, Legendre form.

Equation 3.8 is used in the RMS program that is described is Chapter 6. However, Equation 3.7 is used to demonstrate the RMS concept throughout the report due to its simpler form.

Defining the stress parameter, $\Omega = f_o E\left[S^m\right]$, Equation 3.7 becomes:

$$D = \frac{TB^m \Omega}{A} \tag{3.9}$$

When the damage is greater than or equal to one, failure is usually assumed to occur. Laboratory tests have shown wide variation in the actual cumulative damage at failure. Defining the damage at failure as Δ_f, the above equation can be rewritten as:

$$T = \frac{\Delta_f A}{B^m \Omega} \tag{3.10}$$

We have reviewed the Wirsching's fatigue equation with loading based on Weibull model. There are other methods currently used to calculate Ω including a deterministic method, a spectral approach and etc. [Wirsching, 1983]. Since the Weibull model uses only two variables to describe loading, it provides a very concise format. Whenever the loading information (Weibull parameter ε and extreme stress S_o) is available, the fatigue life can be computed easily. Unfortunately the two loading parameters are very difficult to obtain. This problem is addressed in the following section.

3.2 Fatigue Life Estimation of Repaired Cracks

When a repair is made, a combination of three things can occur:

1. a change in the S-N curve designation of a location due to modifications such as welding;
2. a change in the alternating stress level of a location due to change in geometry; and/or
3. a change in component thickness (thus alternating stress level) due to the addition of a thicker insert plate or doubler.

To estimate repair lives and compare repair alternatives, these three changes must be accounted for.

To examine how the fatigue equation (3.7, 3.8) can be used to evaluate repairs, consider a crack discovered in T_1 years that developed due to fatigue. Assuming a Weibull parameter and curve designation, the extreme stress range (once in a life time) required to produce the failure may be determined.

Due to the many assumptions involved, this stress range is only useful when used on a comparative basis. For example, consider a crack originating at a cutout corner (C class, m_1=3.5, log A_1=14.03, single slope approximation) in the side shell is discovered in T_1 years. Assume the Weibull parameter ε is equal to 0.9. As will be discussed shortly, the Weibull parameter is nearly independent of the repair life. In other words, assuming 0.9 will make very little difference from assuming 1.0 or other values.

The calculated peak Weibull stress range to cause failure (Δ_f=1) based on the mean S-N data and no uncertainty (B=1) is:

$$S_1 = \frac{(\ln (f_o T_1))^{1/\varepsilon}}{B} \left\{ \frac{\Delta_f A_1}{f_o\, T_1\, \Gamma\left(\frac{m_1}{\varepsilon} + 1\right)} \right\}^{1/m_1} \tag{3.11}$$

The average frequency f_0 of the stress cycles is known to be a constant 2.5×10^6 cycles/yr for the wave loading on ship structure. It can be verified by the following calculation assuming 70 percent ship operation and an average wave encounter period of 9 seconds:

$$\begin{aligned} f_o &= 0.70\left(\frac{1 \text{ cycle}}{9 \text{ sec}}\right)\left(\frac{365 \text{ days}}{1 \text{ year}}\right)\left(\frac{24 \text{ hrs}}{1 \text{ day}}\right)\left(\frac{60 \text{ min}}{1 \text{ hr}}\right)\left(\frac{60 \text{ sec}}{1 \text{ min}}\right) \\ &= 2.5 \times 10^6 \text{ cycles / yr} \end{aligned} \tag{3.12}$$

If this crack is then ground out and welded up, the S-N curve degrades to, say, D class (m_2=3.0, A_2=3.99e12). The Weibull parameter remains the same as 0.9 by assuming the ship will continue service in the same route. The assumed value of the Weibull parameter will be shown to be almost independent of the estimated fatigue life T_2 in the next section.

All the data needed to computed T_2 are available except the extreme stress S_2. Apply S_1 and S_2 to the equation of the Weibull extreme value [Ochi, 1990]:

$$S_1 = \frac{1}{\delta}(\ln \frac{N_{T1}}{\alpha})^{1/\varepsilon} \tag{3.13}$$

$$S_2 = \frac{1}{\delta}(\ln \frac{N_{T2}}{\alpha})^{1/\varepsilon} \tag{3.14}$$

Where α = risk parameter,

$N_{T1} = f_0 \times T_1$, number of cycles in the first fatigue life,

$N_{T2} = f_0 \times T_2$, number of cycles in the expected repair life.

Take $\alpha = 1\text{-}e^{-1}$, so S_1 and S_2 will be the characteristic largest value. Combine Equation 3.13 and 3.14, and an expression for S_2 can be obtained:

$$S_2 = S_1 \left(\frac{\ln \frac{N_{T2}}{1-e^{-1}}}{\ln \frac{N_{T1}}{1-e^{-1}}} \right)^{1/\varepsilon} \tag{3.15}$$

Possibly the repair is done not only by veeing and welding but also by adding an extra reinforcing component like a lug. The stress around the cutout opening will be reduced due to the new installed component.

Define a stress reduction factor K_{srf} as the stress level after repair divided by the original stress level (see Chapter 5 for more details). S_2 needs to be multiplied by this factor, whenever a geometry modification is carried out. S_2 is modified by the following equation to correct for changes in stress level due to a geometry modification or component thickness change in the repaired detail:

$$S_2{}' = S_2 K_{srf} \left(\frac{t_{original}}{t_{repair}} \right)^n \tag{3.16}$$

t = Thickness of the repaired and original detail

n = Factor which is dependent on the dominant stress direction

Since typically we deal with Mode I cracking (resulting from tensile stress), n will equal one in most cases.

The new mean life to failure T_2 (D_f=1) may be estimated by solving the following by iteration for T_2 :

$$T_2 = \frac{\Delta_f \, A_2 \, [\ln(f_o T_2)]^{(m_2/\varepsilon)}}{f_o \, (B \, S_2')^{m_2} \, \Gamma\left(\frac{m_2}{\varepsilon} + 1\right)} \tag{3.17}$$

Since the veeing and welding may bring potential defects on the weld, the repaired detail generally has a shorter life than the original. For the CSD used in this case, the repair of veeing and welding without a geometry modification will give a result as follows:

T_1 (years)	5.0	10.0	15.0	20.0
T_2 (years)	2.6	4.7	6.6	8.4

Note that this approach is suitable only for those cracks caused by fatigue. It assumes that there is neither faulty material nor poor construction like incorrect welding procedures, incomplete welding, material defects and misalignment problems.

In short, using the fatigue equation (3.7, 3.8) to compute the expected mean repair life, four sets of variables are needed:

- S-N data of the detail (m_1, A_1)
- First fatigue life of the detail (T_1)
- S-N data of the repaired detail (m_2, A_2)
- Stress reduction factor (K_{srf})

Whenever the above four sets of information are available, then the expected repair life can be computed.

3.3 Load Considerations

3.3.1 Weibull Loading Model

To evaluate a component for fatigue, the alternating stress level must be determined. The effect of mean stress can generally be ignored due to its generally small magnitude and small influence on the fatigue strength of steels [ISSC, 1988, 1991]. Several models can be used to represent the long term stress range, including wave exceedance diagrams, spectral methods, the Weibull model and the Nolte-Hansford model. A Weibull model to represent the long term distribution of cyclic stress ranges is used for the RMS due to its relative simplicity and general applicability.

Using the Weibull model, the alternating stress in ship structure is represented by:

$$F(S) = \Pr(s > S) = \exp\left(-\left(\frac{S}{\delta}\right)^{\varepsilon}\right) \quad (3.18)$$

F(S)	=	Probability that stress range S is exceeded
ε	=	Weibull shape parameter
δ	=	Weibull scale parameter

The scale parameter δ may be related to the stress range and the return period N_O by:

$$\delta = \frac{S_o}{(\ln N_o)^{1/\varepsilon}} \quad (3.19)$$

S_0 is the alternating stress that is exceeded on an average of once every N_0 cycles (design life or actual life in cycles). So now we have a one parameter distribution represented by:

$$F(S) = Pr(s > S) = \exp\left(-\left(\frac{S}{S_o}\right)^{\varepsilon} \ln N_o\right) \tag{3.20}$$

Defining N as the number of stress variations of N_0 that exceed S this equation may be expressed as:

$$S = S_o\left(1 - \frac{\log N}{\log N_o}\right)^{\frac{1}{\varepsilon}} \tag{3.21}$$

The Weibull shape parameter ε will vary with the environment (trading route, sea conditions) and the response of the ship structure to the environment. Specifically, ε will vary with ship length, ship type, location within the ship and the trading route under operation. For crude carriers and cargo ships, ε is typically between 0.7 and 1.3 [Munse, 1981]. See Figure 3.1 for illustration of its influence on the stress range exceedance diagram. It is currently assumed to be 0.9 in the RMS. This assumption is justified in the next section.

3.3.2 Justification of Weibull Parameter Assumption

In the approach of simplified fatigue analysis for repaired CSD, the Weibull parameter has been assumed as 0.9. This assumption needs to be justified. Likewise we want to know how sensitive the fatigue life is to different values of Weibull parameter. In order to find this out, a fatigue crack in a longitudinal cutout radius (corner) is chosen as the sample to be evaluated. Six repair alternatives are considered from simply veeing and welding to adding a lug (see Figure 3.2). Each repair will have effects on increasing or decreasing fatigue strength (S-N data) or a stress level as listed in follows:

Repair	S-N Curve or Stress Changes
1 veeing and welding	degraded from C to E
2 veeing and welding plus treatment	degraded form C to D
3 Inserting a new plate	No change of S-N curve or stress level
4 Adding a lug plus repair 1	Stress is lowered, degraded from C to E
5 Adding a lug plus repair 2	Stress is lowered, degraded from C to D
6 Adding a lug plus repair 3	Stress is lowered, no change of S-N curve

(The stress is reduced to 70% hypothetically for repair #4, #5 and #6.)

In the first case, a crack is assumed to have an original fatigue life of 10 years. The expected fatigue lives of the repaired detail are calculated using different values of Weibull parameters from 0.7 to 1.2. The result is shown in Figure 3.3. From Figure 3.3, we can see that the different values of Weibull parameter give very little difference on the expected repair lives. Then the second case of sensitivity test is carried out by assuming the crack has an original fatigue life of 15 years (Figure 3.4). This case gives a very similar result as the first case. It also shows that the expected repair lives are not sensitive to the Weibull parameter.

The two tests show that while higher Weibull values are assumed, the extreme stress (once in a life time) reduces in such a way that they together contribute a constant amount of damage. The result is amazingly consistent, that is, no matter what value is used the approach produces very close results. In the repair #3 of both cases, the error is as low as 0%. Therefore, the conclusion is made as that the assumption of Weibull value does not influence repair lives significantly. Since the load computation is the most tedious part of a fatigue analysis, these new finding completely eliminate it. This enables a repair life estimation to be done within a minimum time.

3.3.3 Stress Reduction due to Geometry Modification

Fatigue is dependent on the local stress in a CSD. The local crack opening stress may be estimated by detailed finite element analysis. For a ship CSD, the loadings may be broken up into longitudinal stress due to hull bending (vertical and horizontal), shear (vertical), and net pressure. For a complete description of the stress reduction factors from a finite element analysis model, each of these load cases should be applied independently to the part. The results from each of these analyses can then be used to create a table of stress reduction factors (SRF). The SRF is function of the detail configuration, the location within the detail, and the applied stress direction.

While the geometry of a detail is modified due to a repair, we have a change in stress level at the crack location. We can define this change in stress level as a stress reduction factor (K_{srf}):

$$K_{srf} = \frac{S_2}{S_1} \tag{3.22}$$

where S_1 is the hotspot stress before repair.

S_2 is the hotspot stress at the same location after a certain repair.

Considering a ship structure as a linear system, the hotspot stress before repair S_1 can be decomposed into four components as follows:

$$S_1 = S_V + S_H + S_P + S_S \tag{3.23}$$

where S_V = stress due to vertical bending

S_H = stress due to horizontal bending

S_P = stress due to net pressure

S_S = stress due to shear force

After the detail has been repaired by reinforcing, the stress levels of these four components will be lowered independently. In an overall sense S_1 will drop to S_2 by the factor of K_{srf} as the following equation:

$$\begin{aligned} S_2 &= k_V S_V + k_H S_H + k_P S_P + k_S S_S \\ &= S_1\left(\frac{S_V}{S_1}k_V + \frac{S_H}{S_1}k_H + \frac{S_P}{S_1}k_P + \frac{S_S}{S_1}k_S\right) \\ &= S_1(R_V k_V + R_H k_H + R_P k_P + R_S k_S) \\ &= S_1 K_{srf} \end{aligned} \tag{3.24}$$

The overall stress reduction factor K_{srf} for the modified detail can be written concisely as:

$$K_{srf} = \sum_{i=1}^{n} R_i k_i \tag{3.25}$$

i	=	Load case number
n	=	Total number of load cases
k_i	=	Stress reduction factor for load case i
R_i	=	Load ratio for load case i at the ship location under study.

This linear combination is valid only if stress reduction factors are defined normal to the crack direction.

Ri is dependent on location. Depending on the location of the detail within the ship, the effect of these stress reduction factors will vary. For example, around the waterline location of the ship, the stress due to vertical bending is minimal (close to the neutral axis) and the stress due to external pressure is very high (wave loading). Therefore, to compare the stress levels at various locations within several repair alternatives, it would be ideal to develop a load ratio as a function of the location within a ship.

The process of identifying R_i through wave spectrum and global structural analysis is extremely tedious. This development of the RMS will assume that cyclic pressure is the dominant load for CSD fatigue cracks. In other words, we assume Rp is closely equal to one and the other three load ratios are zero. Equation 3.25 becomes:

$$K_{srf} = k_P \tag{3.26}$$

With this assumption K_{srf} will be independent of crack location. While making this assumption, the stress reduction factors in RMS will be more effective on the details subjected to cyclic wave or internal pressures. In other words, they are suitable for those details under pressure load like side shell or forepeak structure.

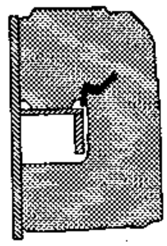
In Chapter 5, we calibrate K_{srf} for a variety of repair alternatives using finite element analysis. When new details are analyzed by finite element methods or by testing, results can be stored in a tabular format for immediate use in the evaluation of repairs.

3.4 Example of Repair Alternative Evaluation

A failure example in a longitudinal cutout will be analyzed to illustrate how this evaluation process proceeds. A crack in the cutout radius is assumed to be discovered at a ship life of 10 years (T_1). As a temporary repair, the stress concentration factor of approximately 9 for the sharp crack can be reduced to approximately 3 simply by drilling a hole at the crack tip [ISSC, 1991]. However that is not a formal repair. Five repair alternatives are evaluated here.

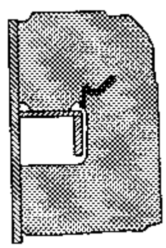
Repair 1 Vee and Weld

The geometry of this detail has not been modified and the loadings are unaffected. As a result, the stress at the crack location will remain relatively unchanged except for the addition of the weld. The material degradation due to welding is accounted for by modifying the S-N curve from C to E class. Following the computing procedures of the previous section, the result of the expected repair life is about 3.84 years only.



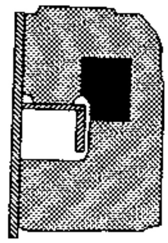
Repair 2 Vee and Weld Plus Post Weld Improvement

This repair is almost the same as Repair 1. Since the weld surface is improved, the material degradation due to welding is accounted for by modify the S-N curve from C to D class. Following the computing procedures of the previous section, the result of the expected repair life is about 4.66 years only.



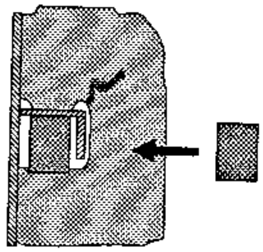
Repair 3 Insert a New Plate:

The geometry of this detail has not been modified, but the insert plate thickness may be different from the original plate. At the crack location, the expected life of this repair is assumed to be equal to that of the original, that is, 10 years. In case that the plate thickness t is modified, the new stress range should be estimated by Equation 3.16. A better repair can be obtained. Notice that two new hot spots are introduced by the weld around the inserted plated. At the weld locations, a combination of a stress concentration factor increases due to the change in plate thickness and a change in the S-N curve due to the addition of the weld occurs. Therefore the inserted plate should be carefully configured to avoid new hot spots.



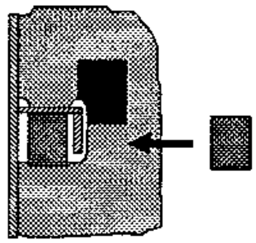
Repair 4 Vee & Weld Plus Add a Lug:

In this case the geometry has been modified so that we have a change in stress level plus a change in S-N curve designation at the crack location. The change in stress level is determined by a stress reduction factor. The stress reduction factor can be sound in Chapter 5. Notice that the S-N curve has been degraded at the lug weld location and at the location of the crack, too. Each of these locations should be evaluated separately.



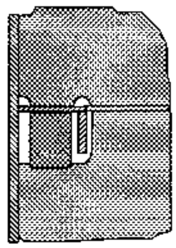
Repair 5 Insert Plate Plus Add a Lug:

In this case the geometry has been modified so that we have a change in stress level plus a change in S-N curve designation at the weld locations. There is no change in the S-N curve at the original crack location, but possibly a change in plate thickness of the inserted plate. Evaluation continues as for Repair 4.



Repair 6 Add a Bracket Plus Repair 5:

In this case the geometry has been modified beyond repair 4 with the addition of brackets. Evaluation continues as for Repair 5.



A simplified approach to the estimation of the fatigue life of repair alternatives has been outlined and demonstrated for a typical side shell structural detail. Depending on the data available, some required information might be missing to estimate the repair life. The RMS should report this missing data and allow for easy addition of any new results to the database.

3.5 Fatigue Reliability

Significant uncertainty exists in the factors of fatigue damage expressions in Section 3.1. For example, cycles to failure data typically have coefficients of variation ranging from 50 to 150%. Similarly, the process of computing fatigue stresses from oceanographic data contains a sequence of several operations that can produce both random and systematic errors in the stress estimates.

This section reviews a reliability model for a ship CSD subjected to a potential fatigue problem. The variables in the damage expression are considered as lognormally distributed random variables. The resulting reliability format is useful for evaluating the probability of failure of a ship structural detail.

Miner's rule states that failure under irregular stress ranges occurs when fatigue damage $D \geq 1$. But random fatigue experimental results have suggested that it is appropriate to describe fatigue failure more generally as

$$D \geq \Delta \tag{3.27}$$

where Δ = a random variable denoting damage at failure.

Δ is defined as a random variable in order to quantify the inaccuracies associated with using a simple model to describe a complicated physical phenomenon. Uncertainties in fatigue strength, as evidenced by scatter in S-N data, are accounted for by considering A to be a random variable (with m = a constant). Inaccuracies in the process of estimating fatigue stresses from oceanographic data are described by the random variable B.

Let T denote time to fatigue failure. Letting $D = \Delta$, the basic damage expression of Equation 3.9 can be expressed in terms of time to failure,

$$T = \frac{\Delta A}{B^m \Omega} \tag{3.28}$$

Because Δ, A, and B are random variables, T also is a random variable. The probability of fatigue failure is defined as:

$$P_f = P(T \leq T_S) \tag{3.29}$$

where T_S = service life of the structure.

In Wirsching's reliability model [Wirsching, 1983], each random variable is assumed to have a lognormal distribution. Employing mathematical properties of lognormal variables, an expression for P_f can be derived as,

$$P_f = \Phi(-\beta) \tag{3.30}$$

where $\Phi(\cdot)$ is the standard normal distribution function, and β is defined as the safety index.

$$\beta = \frac{\ln\left(\tilde{T}/T_s\right)}{\sigma_{\ln T}} \tag{3.31}$$

$\tilde{T}$ is the median value of T and equal to,

$$\tilde{T} = \frac{\tilde{\Delta}\,\tilde{A}}{\tilde{B}^m\,\Omega} \tag{3.32}$$

The tildes indicate median values. Also, the standard deviation of log T is

$$\sigma_{\ln T} = \sqrt{\sigma_{\ln\Delta}^2 + \sigma_{\ln K}^2 + m^2\sigma_{\ln B}^2} \tag{3.33}$$

$$\sigma_{\ln T} = \sqrt{\ln\left(1+C_\Delta^2\right)\left(1+C_K^2\right)\left(1+C_B^2\right)^{m^2}} \tag{3.34}$$

where the C's are the coefficients of variation of each variable.

It should be emphasized that, because of the lognormal assumption for Δ, A and B, and because of poor definition of distributions in the critical tail areas resulting from lack of data, computed values of P_f do not necessarily provide precise estimates of risk. Values of P_f are useful in a relative sense.

The principal reason for using the lognormal format is that fatigue life, T, has an exact lognormal distribution when Δ, A, and B are lognormal. It results a relatively simple closed form and exact expression for P_f. A complicated probability problem is created when any other distribution is used for any of the variables. Moreover, it has been demonstrated that the lognormal is a valid model for a wide variety of structural design variables. The lognormal model has been shown to provide a good fit to data on Δ. It is also considered to be reasonable for B and A.

For a reliability analysis it is necessary to specify the median and the coefficient of variation of A, B, and Δ. The values of $\tilde{K}$ and C_K is obtained from the S-N data. For Δ describing the modeling error associated with Miner's rule, the values of $\tilde{\Delta} = 1.0$ and $C_\Delta = 0.3$ are widely used. The variable B is used to quantify the modeling error associated with assumptions made in the stress analysis and the description of fatigue strength. Several sources can contribute to the bias B. Wirsching uses five contributors. The frequently used values of the medians and COV's of the five are listed [Wirsching, 1984]:

	Bias	COV
B_M = Fabrication and assembly operations	0.9-1.3	0.1-0.3
B_S = Sea state description	0.6-1.2	0.4-0.6
B_F = Wave load prediction	0.6-1.1	0.1-0.3
B_N = Nominal member loads	0.8-1.1	0.2-0.4
B_H = Estimation of hot-spot SCFs	0.8-1.2	0.1-0.5

For the fatigue life estimation of a repaired detail, the load information is directly derived from the load of the original fatigue life. Therefore only B_M, B_s and B_F are needed. Using these three bias factors the following representation of B is obtained:

$$B = B_M B_S B_F \tag{3.35}$$

Assuming the ship will continue serve on the same route, B_s and B_F may be further eliminated. Assuming that each random variable is lognormally distributed the median and the COV of B are respectively:

$$\tilde{B} = \prod_i \tilde{B}_i \tag{3.36}$$

$$C_B = \sqrt{\prod_i (1 + C_i^2) - 1} \tag{3.37}$$

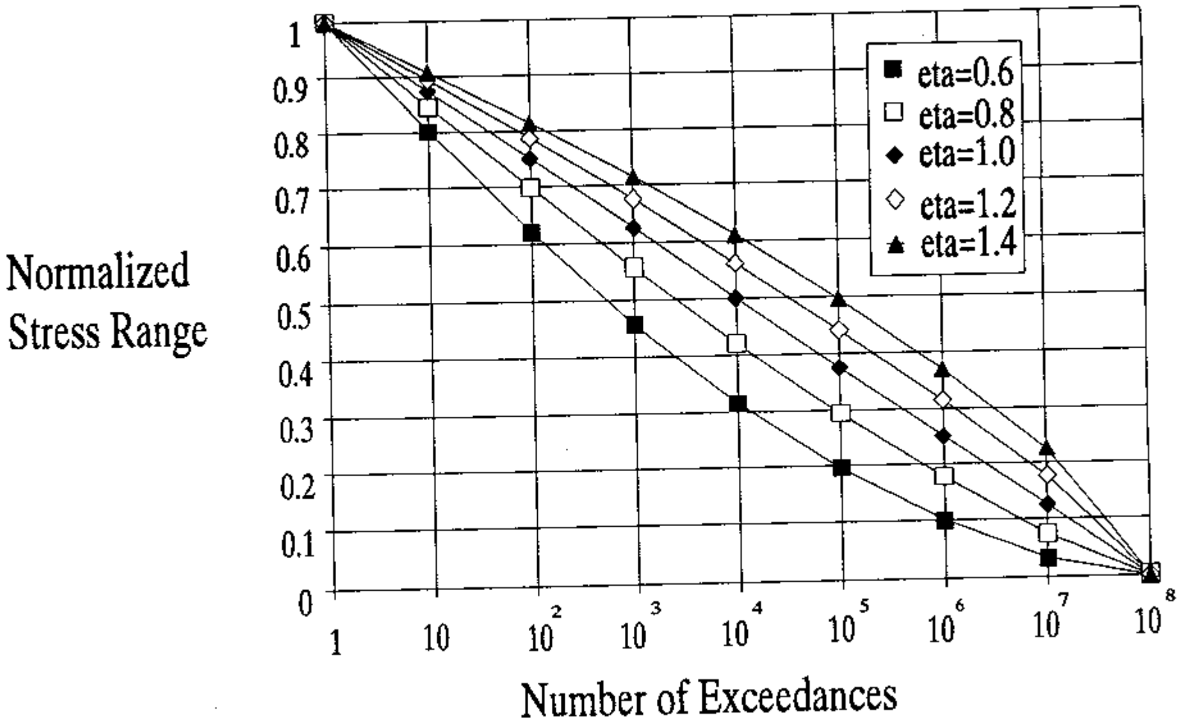


Figure 3.1: Five Weibull distributions with different values of Weibull parameter shown on a stress-range exceedance diagram.

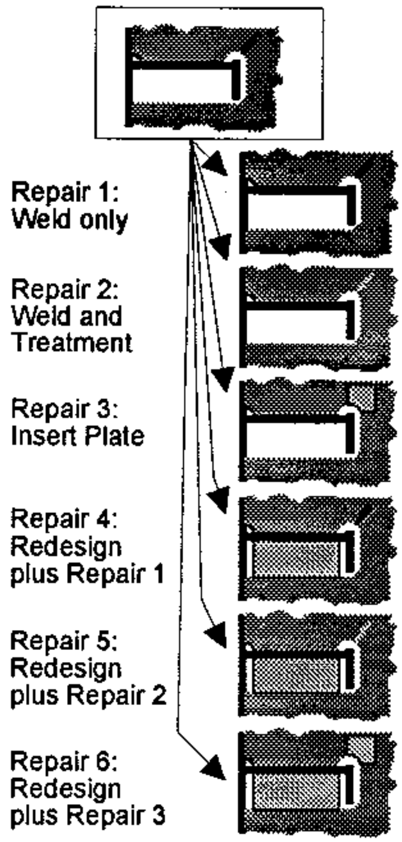
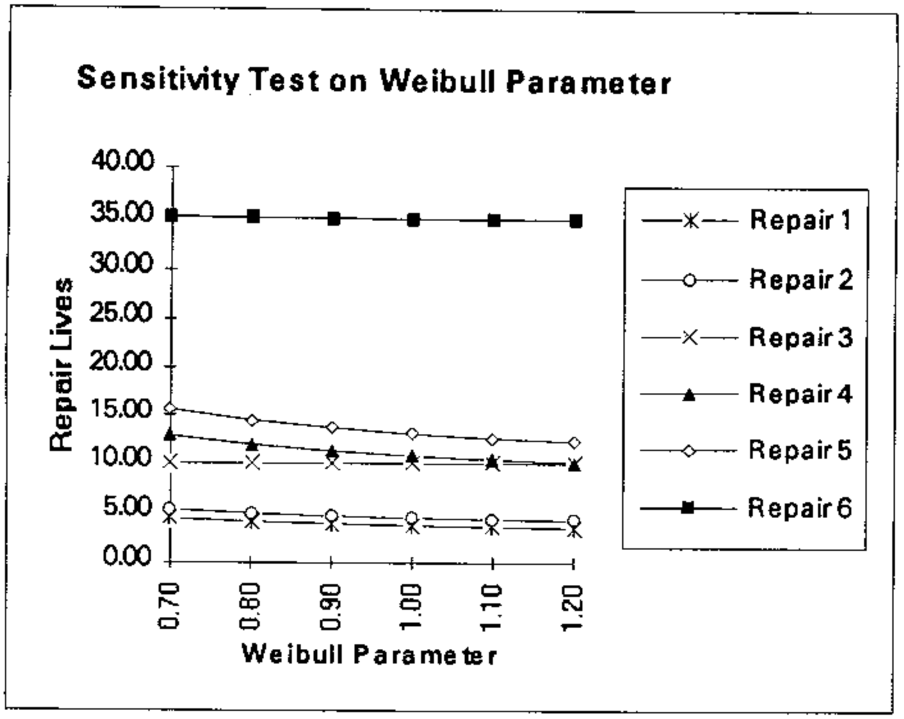
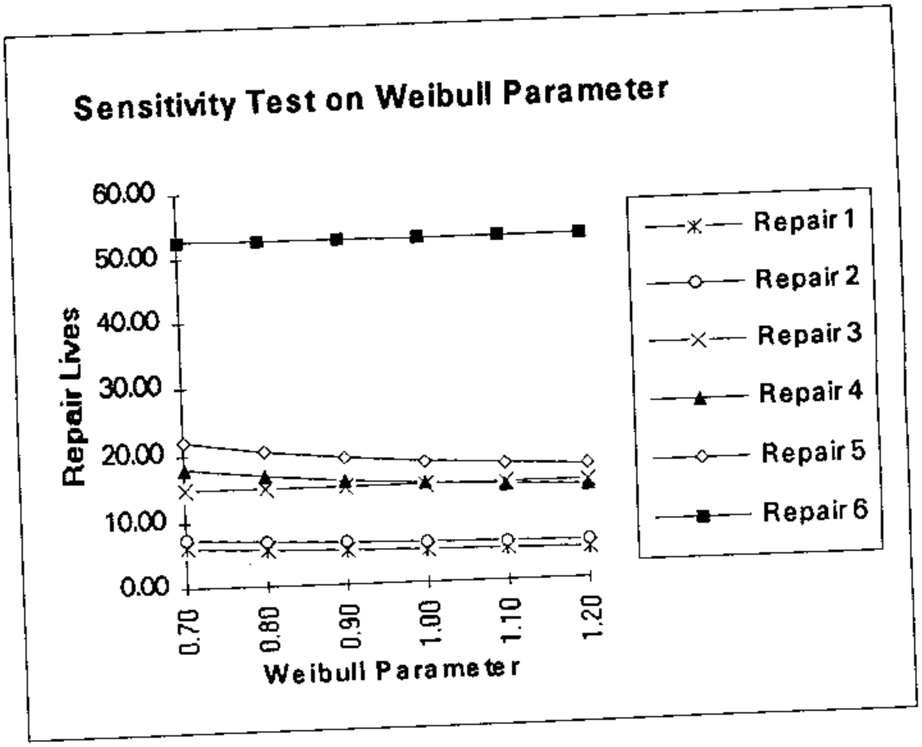


Figure 3.2: Six repair alternatives on a fatigue crack in cutout radius.



	Sensitivity Test of Weibull Parameter					
	(Fatigue Crack in Cutout Radius at age of 10 years old)					
			ε			
(Years)	0.70	0.80	0.90	1.00	1.10	1.20
Repair 1	4.37	4.06	3.84	3.67	3.54	3.44
Repair 2	5.30	4.93	4.66	4.46	4.30	4.18
Repair 3	10.00	10.00	10.00	10.00	10.00	10.00
Repair 4	12.83	11.91	11.25	10.76	10.38	10.08
Repair 5	15.58	14.46	13.66	13.06	12.60	12.23
Repair 6	35.16	35.12	35.09	35.07	35.05	35.03

Figure 3.3: Test the sensitivity of Weibull parameter (case 1).



	Sensitivity Test of Weibull Parameter					
	(Fatigue Crack in Cutout Radius at age of 15 years old)					
			ε			
(Years)	0.70	0.80	0.90	1.00	1.10	1.20
Repair 1	6.19	5.75	5.43	5.20	5.01	4.87
Repair 2	7.51	6.98	6.60	6.31	6.09	5.91
Repair 3	15.00	15.00	15.00	15.00	15.00	15.00
Repair 4	18.16	16.86	15.93	15.23	14.69	14.26
Repair 5	22.05	20.46	19.33	18.48	17.83	17.31
Repair 6	52.72	52.66	52.62	52.58	52.56	52.53

Figure 3.4: Test the sensitivity of Weibull parameter (case 2).

CHAPTER 4. S-N DATA

To calculate the expected fatigue life of a given repair, both load and strength data are needed as described in the previous chapter. The load can be described by a Weibull load model. The fatigue strength can be described by an S-N curve. In this chapter we summarize explain the traditional ways of obtaining S-N data for a Critical Structural Detail (CSD). In addition we will introduce a new method to calibrate the S-N data for repaired CSD. These data are a necessary input in estimating fatigue lives.

4.1 S-N Curve

Fatigue refers to the failure of materials under repeated actions of stress fluctuation. The loads responsible for fatigue are generally not large enough to cause material yielding. Instead, failure occurs after a certain number of load or stress fluctuations.

Fatigue strength is therefore not represented by a single stress value but by a curve on a chart of stress range related to number of cycles. Ideally to obtain the fatigue strength of a CSD, a prototype of the CSD needs to be tested under different values of constant stress ranges. Due to the size and complex geometry of a CSD, it is almost impossible to perform such tests to obtain the fatigue strength information.

Therefore laboratory specimens are made and tested with alternating loads. The relation between the stress range, S, and number of cycles to failure, N, is plotted as a curve. This curve is called S-N curve and is assigned a letter such as B, C, D, E, F, F2, G, W or others (Figure 4.3). Different curves represent specimens of different configuration. A ship structural detail can be matched to the S-N curve of a laboratory specimen if it has a similar geometry and loading condition to the specimens. Different locations within a detail are assigned an S-N curve that represents the fatigue characteristics of that location.

An indication of the relationship between a ship structure detail and a laboratory fatigue specimen is given in Figure 4.1. The shown fatigue specimen (right side in Figure 4.1) is classified into the class F by the U.K. Department of Energy (U.K. Den) [Chen, 1992]. Since the ship structural detail shown in the left side of the figure has a similar geometry and loading condition as the specimen, the detail can be assigned a S-N curve of Class F. Naval architects have been investigating which S-N curve should be assigned to the corresponding critical location in a CSD [Chen, 1992].

There is an amount of judgment involved in the assigning of an appropriate S-N curve to a critical location in a CSD. Work on matching S-N curves to ship structural detail has been explored [Chen, 1992]. The study assigned U.K. DEn S-N curves to some critical location in a variety of ship CSDs representing current design and shipyard practice of tanker structure. Figure 4.2 shows some results from that study.

Beside U.K. DEn curves, there are also S-N curves that are developed by other organizations. Department of Civil Engineering of the University of Illinois has developed sets of S-N data based on small specimens [Munse, 1983]. Since significant amount of work has been done by ABS in the classification of CSDs for the use of U.K. DEn S-N curves, we adopted the classification of CSDs by ABS. Therefore the U.K. DEn curves are used in the RMS.

Figure 4.3 summarizes the design S-N curves associated with these designations. S-N class designations closer to "A" in the alphabet represent more durable locations than those designated by a subsequent letter. These curves, which represent the mean data (for design purposes) of log N, may be described by:

$$\log N = \log A - m \log S \tag{4.1}$$

N	=	Predicted number of cycles to failure under stress range S
A	=	Life intercept
S	=	Applied constant amplitude stress range
m	=	Inverse slope of S-N curve

The U.K. DEn specifications provide tables relating to selection of S-N curves for any given structural detail situation.

It has been observed that the logarithm of N is approximately normally distributed at a particular stress range [Wirsching, 1983]. Others have fit different distributions. In the log-normal approach, the mean S-N curve is found by performing a linear regression analysis, minimizing the error in log N using the method of least squares with log S as the independent variable. The large variance in the number of cycles is primarily due to variance in the weld geometry and weld imperfections.

Some fatigue analyses are based on the S-N curve which is two standard deviations below the mean to ensure a safer design. This means that two standard deviations are deducted from mean S-N curves to be on the safe side of test results (See Figure 4.4), that is, 97.5% survival S-N curve is obtained. In RMS, the mean S-N curves are used. As a result, the estimated fatigue lives in RMS will be mean (or expected) fatigue lives.

There is a size effect associated with these curves [Chen, 1992]. To account for this, Equation 4.1 may be modified to the following for all types of welded structure except for butt welds dressed flush and low local bending across the plate thickness:

$$\log N = \log A - \frac{m}{4} \log\left(\frac{t}{22}\right) - m \log S \qquad (4.2)$$

The variable t is the thickness in millimeters through which a crack will grow (e.g., plate thickness).

The strength and type of steel typically have only a small negligible effect on the fatigue life of a particular weld detail in the longer life regime, even if tests on the un-welded base plate indicate material dependence. The welding process also does not typically have a large effect on the fatigue strength, unless a unique discontinuity is produced. This material independence is more evident in large-scale defect-dominated specimens than in some small-scale tests [Fisher & Dexter, 1993]

It is worth noting that S-N performance is also affected by the environment. When steel is subjected to cyclic stresses while in contact with a corrosive environment like sea water, the fatigue strength may be reduced as compared with the fatigue strength for the same number of cycles in air. In tankers, the rules of some class societies now require coating in ballast tanks, so only cargo tanks without coating will potentially suffer this corrosion fatigue.

There are two distinct regions in Table 4.1. For cycles $N>10^7$ there is a change in slope to take account of the corrosion effect. There is some controversy over the actual effect of sea water and cathodic protection on these curves; however, the RMS will allow the S-N curve data to be modified to the form desired by the user. For unprotected steel in sea water, a fatigue strength is assumed to be reduced by a factor of 2.0 [Chen, 1992].

4.2 Nominal or Hot Spot S-N Curves

There are two approaches to define S-N curves: nominal stress approach and hot-spot stress approach. The nominal stress approach is used in the RMS. However it is worthwhile to explain both approaches here. The concept of the hot-spot stress approach will be helpful to the stress recovery which will be discussed in the next chapter.

The nominal stress approach separates ship structural details into categories having similar fatigue resistance. A fatigue curve must be generated for each of these categories. The stress concentration effects associated with the shape of the weld and the local geometry of the detail is an integral part of the fatigue resistance. Therefore, the loading is generally characterized in terms of the nominal stress in the plate remote from the weld detail. In ship primary components like longitudinals, the nominal stress can be conveniently obtained from standard strength of materials equations using member forces and moments from a global analysis. In most ship internal structural components made from continuous plating, the nominal stresses are typically obtained from a finite element analysis. Use of simple strength of material equations to determine stresses makes the approach straightforward. However the proper definition of the nominal stresses may become a problem in regions of high stress gradients.

The second approach is called hot-spot stress approach. The hot-spot stress approach uses a reduced number of S-N curves but involves more complicated analyses [Schulte-Strathaus and Bea, 1993]. In the hot-spot stress approach, S is equal to the 'hot-spot" stress or the product of the nominal stress times the stress concentration factor (SCF). This definition of S results in one single base-line S-N curve that can replace several nominal S-N curves for details with welds. Theoretically, the base-line S-N curve can represent the fatigue resistance of a range of details. An S-N curve associated with butt welds or fillet welds in a nominal stress field is chosen as the base-line S-N curve. In this case, the effect of the local stress concentration at the weld toe is included in the S-N curve and therefore the SCF includes only effects associated with global geometry.

One problem with the hot-spot stress approach is that the stress gradients are very steep in the vicinity of the weld. Because of the high gradients, the maximum stress computed or measured will be sensitive to the mesh size or the gage length. Because of this mesh sensitivity and because the effect of local stress concentration to be separated from the SCF, the hot-spot stress must be defined in an arbitrary way, e.g. the stress at a certain distance from the weld toe. Unfortunately, there are various and ambiguous definitions of hot-spot stress. The hot-spot stress range is either measured using strain-gages or calculated using finite element analysis. There are various base-line S-N curves corresponding to different definitions of the hot-spot stress. The base-line S-N curve discussed above is for structural details with welds. For plain metal (e.g. cutouts) other higher base-line curves should be used [Schulte-Strathaus and Bea, 1993].

4.3 S-N Curves of Repaired Details

As discussed in Chapter 3, three sets of information are needed to compute the expected repair life. The first set, S-N data of the structural detail, can be obtained as shown in Figure 4.2 or from a past study [Chen, 1992]. The second set, the fatigue life of the detail, is the time interval from the delivery of the ship to the time the detail fails.

The third set, S-N data of the repaired detail, is not readily available. Only when the repair is done by inserting a new plate, the S-N curve for the original crack site is sure to be the same curve as the original one, since the geometry and the material of the detail stay the same. In the following, we will review two examples of the unavailability of S-N data.

In the first example, consider a critical location at a cutout radius in a newly constructed transverse web, a C class of S-N curve is assigned according to the study of classification of CSDs [Chen, 1992]. After this cutout radius has been in service for a few years, a fatigue crack may develop there. The crack can be repaired by veeing and welding. To estimate the fatigue life of this repair, the S-N class needs to be obtained. For a butt-welded plate a D curve should be used. However the repaired web cutout has been in service for a period of time. The integrity of the plate may has been degraded due to possible corrosion, poor weld, used-up fatigue resistance and other factors. In this case a S-N curve lower than D-class should be considered.

In the second example, a fatigue crack is found in a flatbar heel. An F2-class curve is usually assigned to this type of location. While a flatbar heel is repaired by rewelding, it will probably not have the same fatigue strength as F2-class curve since the flatbar is old. The similar argument can be applicable to other repair by rewelding.

To the present time, no experiments have been carried out to designate a S-N class for the above two examples. To address the unavailability of S-N information for repaired CSDs, the RMS developed a new approach to calibrate the S-N curves of repaired structural details. Also, the S-N data in the RMS program is designed to be readily changed by users. Users can update the S-N data input file whenever new information is available.

4.4 S-N Calibration from Historical Inspection Data

The last section described the difficulties in selecting a proper S-N curve for a repaired CSD. The S-N classification of CSDs is mainly based on engineering judgment. A certain class of S-N curve is compared and matched to a critical location of a CSD with similar geometry as fatigue specimens. Inaccuracy may be introduced in this matching process.

Small scaled specimens used in fatigue experiments could be used. S-N data based on such experiments are used widely by naval architects to represent the fatigue strength of full-scale ship CSDs which are larger than the fatigue specimens.

To address this problem, an approach is developed to establish a S-N classification of repaired CSDs.

4.4.1 RMS Procedure to Calibrate S-N Data

In the RMS approach, we propose a full scale fatigue experiment for repaired CSDs by utilizing historical inspection data to calibrate the fatigue classification of repaired CSDs. The full scale experiments will not be carried out in the laboratory. Instead, we consider the CSDs in in-service ships as fatigue specimens. With gathered historical repair data, the S-N curves of particular repaired CSDs can be developed.

This approach is developed as follows. Assuming a fracture in a longitudinal cutout radius was found while the ship was T_1 years old, we can see that this fracture took T_1 years to develop. The repair was done then by grinding out and re-welding according to the decision of a repair engineer. Unfortunately the repaired fracture re-cracked again in T_2 years. Now the problem is defined as to establish an S-N class for the re-welded circular corner of a longitudinal cutout.

The S-N class of the circular corner of a longitudinal cutout can be matched by C-class [Chen, 1992]. T_1-year loads which attack the hotspot of the longitudinal cutout can be modeled by a Weibull distribution. This load information can be computed by following the simplified fatigue analysis approach established in Chapter 3. This load information will be used to predict the next fatigue life of the repaired detail.

After the fracture was repaired by veeing and re-welding, the detail re-cracked in T_2 years. The fatigue damage factor of veeing and welding has accumulated to one within the T_2 years. Assuming the ship continued serving on the same trade route, the Weibull parameter remains the same but the new extreme stress range can be modified according to the length of T_2. At this point we have had enough data to compute the S-N data of the veeing and welding. By using a reversal fatigue analysis process, the S-N intercept value, A, can be calculate (see Figure 4.5). This procedure will be further developed and illustrated in the following sections.

4.4.2 Gathering Historical Inspection Data

The previous section briefly reviews the procedures of establishing S-N class from one set of inspection data. While more historical inspection data are gathered, the mean of the estimated S-N parameters can be computed. It will converge to the real mean value when a large number of inspection data are available and applied. The S-N classification of other types of repairs can be developed by the similar procedures.

To fulfill the objective of finding the S-N information for repaired details, historical inspection data is needed. We need a large number of cracks that failing in T_1 years and some of the repairs fail again after T_2 years.

A large amount of tanker survey reports have been reviewed in a major ship owner/operator's library to locate the necessary data. The reports of two tankers have been found to be suitable for this study because of the large amount of fatigue failures found in the two ships. Both ships were built in 1972. They have experienced a large number of fatigue cracks through their 22-year service lives.

In the case of one ship, a few fatigue cracks were first found in flat bar welds (FBW) on the side shell longitudinals at its age of 9 years. In the following year more cracks were discovered in forepeak tank. Most of these cracks were around cutout radius of horizontal stringers (CRHS). Until the age of 12 years about 50 cracks of this type have been found in the forepeak tank. At 15 years old a thorough survey and repair was carried out. Many more cracks were found this time and most of them were FBW or CRHS cracks. Three hundred and fifty eight FBW cracks and twenty five CRHS cracks were found and repaired. After 4 years, 4 out of 358 FBW cracks and 7 out of 25

CRHS cracks failed again. Having obtained the needed information a 'reversal' fatigue analysis can be performed.

4.4.3 S-N Calibration Case One

The failures in the cutout radius of horizontal stringers (CRHS) are used for the first case of S-N calibration (Figure 4.6). The tanker has 7 horizontal stringers in the forepeak tank and each of them contains 28 cutouts (totally 196 CRHSs). The survey reports show that twenty-five CRHS fatigue cracks have been found within 15 years. After these 25 cracks were repaired by veeing and welding, seven of them re-cracked within 3 years. With these data, one can define the S-N data for the repair of veeing and welding in cutout radius.

The first thing required to solve for is the average (mean) fatigue life of the CRHSs which is denoted as $\mathbf{T_1}$. We will assume $\mathbf{T_1}$ is a random variable with a lognormal distribution based on the fact that it has only positive value. In addition we assume the inspection data obtained are based on high quality inspection, that is , all significant cracks in the tanker are discovered.

Twenty-five failures were found from 196 CRHSs within 15 years, so we get:

$$P(\mathbf{T_1}<15) = F(15) = 25/196 = 0.1276 \tag{4.3}$$

While set $\mathbf{Y} = \ln \mathbf{T_1}$, the above equation becomes:

$$P(\mathbf{Y}<2.708) = 0.1276 \tag{4.4}$$

where $\mathbf{Y}$ has a normal distribution.

In order to find its median value, $\overline{\mathrm{Y}}$, the standard deviation of $\mathbf{Y}$ is needed. Because no such information can be obtained from the survey report, the value of σ_Y = 1.59 is used based on results from a study by Wirsching [1983].

Letting $\mathbf{z} = (\mathbf{Y} - \overline{\mathrm{Y}})/\sigma_Y$, the normal variable **Y** can be standardized, and the above equation 4.4 changes to:

$$\Phi(z) = \Phi\left(\frac{2.708 - \overline{Y}}{1.59}\right) = 0.1276 \quad (4.5)$$

From the table of standard normal distribution, $\Phi^{-1}(0.1488) = -1.1375 = \dfrac{2.708 - \overline{Y}}{1.59}$. Thus $\overline{Y}$ = 4.5166. The medium value of the fatigue life can be obtain as $\overline{T}_1$ = exp(2.985) = 91.52 (years).

Now the mean fatigue life μ_{T1} can be computed using the relationship between mean and median:

$$\mu_{T1} = \overline{T}_1\sqrt{1 + C_{T1}^2} = 324.0(\text{years}) \quad (4.6)$$

where C_{T1} is the coefficient of variation of $\mathbf{T_1}$ and can be obtained from $\sigma_Y = \ln(1 + C_{T1}^2)$.

The mean fatigue life of the CRHS has been estimated using the assumption of lognormal distribution. Similarly, we can assume $\mathbf{T_2}$ is a random variable with a lognormal distribution. From the fact that 7 failures were found from 25 repaired CRHSs within 3 years, we start the calculation with:

$$P(\mathbf{T_2}<3) = F(3) = 7/25 = 0.280 \quad (4.7)$$

Following the same procedures used in solving μ_{T1}, the mean fatigue life of the repaired CRHS was found to be μ_{T2} = 26.83 years.

After μ_{T1} and μ_{T2} have been estimated, we are ready to calibrate S-N data by reversing the procedures of the simplified fatigue analysis developed in Chapter 3. The first mean fatigue life μ_{T1} (or simply denoted by T_1) is 324 years. The other variables in the equation are known also (C class, m_1=3.5, A_1 = 1.08e14, Weibull parameter 0.9, f_o=2.5x10^6 cycles/year, B=1, Δ_f=1). Using the equation 3.4 the extreme stress range S_1 can be computed from:

255

$$S_1 = \frac{(\ln (f_o T_1))^{1/\varepsilon}}{B} \left\{ \frac{\Delta_f A_1}{f_o T_1 \Gamma\left(\frac{m_1}{\varepsilon} + 1\right)} \right\}^{1/m_1} \tag{4.8}$$

The extreme stress range S_1 for 70 years can now be changed to S_2 for 26.83 years according to a long-term extreme value prediction (see Chapter 3). While the extreme stress range S_2 is known and Weibull parameter does not change, we have the load information of the repair, veeing and welding. Our goal is to find out the S-N data for that repair.

Rearranging equation 3.5, we can get the following equation. While T_2 is known as 26.83 years and m_2 is assumed to be 3.0, the S-N intercept 'A_2' can be obtained as:

$$A_2 = \frac{T_2 f_o \, (B \, S_2)^{m_2} \, \Gamma\left(\frac{m_2}{\varepsilon} + 1\right)}{\Delta_f [\ln(f_o T_2)]^{(m_2/\varepsilon)}} = 1.17 \times 10^{12} \tag{4.9}$$

The result indicates that **the veeing and welding of the CRHS crack has an S-N curve very close to class F2** (see Figure 4.3).

4.4.4 S-N Calibration Case Two

The failures in the flatbar weld (FBW) are used for the second case of S-N calibration (Figure 4.7). The tanker has 4240 flatbars in its side shell and longitudinal bulkhead. It also has flatbars on its bottom longitudinals. However due to the possible different loading pattern, the bottom area is neglected. The survey reports show that 358 FBW fatigue cracks have been found within 15 years. After these 358 cracks were repaired by veeing and welding, four of them re-cracked within 3 years. With these data, one can define the S-N curve for the repair of flatbar weld. The procedures are the same as the S-N calibration case in the previous section.

The first thing we need to determine is the average (mean) fatigue life of the FBWs which is denoted as $\mathbf{T_1}$. We assume $\mathbf{T_1}$ is a random variable with a lognormal distribution based on the fact that it has only positive value. Three hundred and fifty eight failures were found from 4240 FBWs within 15 years, so:

$$P(\mathbf{T_1}<15) = F(15) = 358/4240 = 0.0844 \tag{4.10}$$

While set $\mathbf{Y} = \ln \mathbf{T_1}$, the above equation become:

$$P(\mathbf{Y}<2.708) = 0.0844 \tag{4.11}$$

where $\mathbf{Y}$ has a normal distribution.

In order to find its median value, $\overline{Y}$, the standard deviation of $\mathbf{Y}$ is needed. Because no such information can be obtained from the survey reports, the value of σ_Y = 1.59 is used based on results from the study by Wirsching [1983].

Letting $\mathbf{z} = (\mathbf{Y} - \overline{Y})/\sigma_Y$, the normal variable $\mathbf{Y}$ can be standardized. And equation 4.11 becomes:

$$\Phi(z) = \Phi\left(\frac{2.708 - \overline{Y}}{1.59}\right) = 0.0844 \tag{4.12}$$

From the table of standard normal distribution, $\Phi^{-1}(0.0844) = -1.376 = \dfrac{2.708 - \overline{Y}}{1.59}$.

Thus $\overline{Y}$ = 4.896. The medium value of the fatigue life can be obtain as $\overline{T}_1$ = exp(4.896) =134 (years).

Now the mean fatigue life μ_{T1} can be computed using the relationship between mean and median:

$$\mu_{T1} = \overline{T}_1\sqrt{1 + C_{T1}{}^2} = 473.3(\text{years}) \tag{4.13}$$

where C_{T1} is the coefficient of variation of $\mathbf{T_1}$ and can be obtained from $\sigma_Y = \ln(1 + C_{T1}{}^2)$.

The mean fatigue life of the FBW has been estimated using the assumption of lognormal distribution. Similarly, we can assume $\mathbf{T_2}$ is a random variable with a lognormal distribution. From the fact that 4 failures were found from 358 repaired FBWs within 3 years, we start the calculation with:

$$P(\mathbf{T_2}<3) = F(3) = 4/358 = 0.011 \qquad (4.14)$$

Following the same procedures used in solving μ_{T1}, the mean fatigue life of the repaired FBW was found to be μ_{T2} = 400 years.

After μ_{T1} and μ_{T2} have been estimated, one can determine the S-N curve by reversing the procedures of the simplified fatigue analysis developed in Chapter 3. The first mean fatigue life μ_{T1} (or simply denoted by T_1) is known as 473.3 years. The other variables in the equation are known also (F2 class, m_1=3.0, A_1 = 1.23e12, Weibull parameter 0.9, f_0=2.5x10^6 cycles/year, B=1, Δ_f=1).

Using the equation 4.6 the extreme stress range S_1 can be computed. The extreme stress range S_1 for 473 years can now be changed to S_2 for 400 years according to a long-term extreme value prediction (see Chapter 3). While the extreme stress range S_2 is known and Weibull parameter does not change, we have the load information of the repair, veeing and welding. Our goal is to find out the S-N data for that repair.

While T_2 is known as 400.7 years and m_2 is assumed to be 3.0, the S-N intercept 'A_2' can be obtained from equation 4.8 as:

$$A_2 = \frac{T_2 f_o\ (B\ S_2)^{m_2}\ \Gamma\left(\frac{m_2}{\varepsilon} + 1\right)}{\Delta_f[\ln(f_o T_2)]^{(m_2/\varepsilon)}} = 1.04\text{x}10^{12} \qquad (4.15)$$

The result indicates that **the re-welding of the FBW crack has an S-N curve very close to its original strength, i.e. class F2** (see Figure 4.3).

4.4.5 Summary of the Two Calibrations

In summary, the result of the second case indicates that repaired details have very similar S-N classification (but slight lower) to normal details with similar geometry. In other words, while a newly built FBW has an S-N curve of F2-class, a repaired FBW has the same class, too. The slightly lowering can be explained by possible corrosion that continually waste the plate thickness. It may also be explained by that the quality of re-welding is lowered due to a difficult working conditions during staged repairs. This result agrees with our expectations and intuition.

The result of the first case indicates that the C-class of a cutout radius is lowered down to a F2-class after a repair by veeing and welding. This result is not our expectation in that we consider a D-class or E-class would be more likely to be the result. This result may be explained by insufficient data or consistently poor quality repairs. The second case has a total number of FBWs of 2120 and a number of first failures of 358. The first case has only 196 and 25. The first case apparently does not have enough data to produce a realistic result. Therefore we conclude that the approach is valid only when a sufficient number of data is collected.

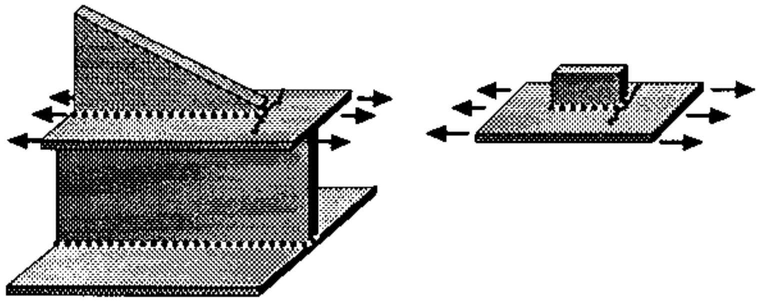


Figure 4.1: A ship structural detail and the corresponding F-class fatigue specimen

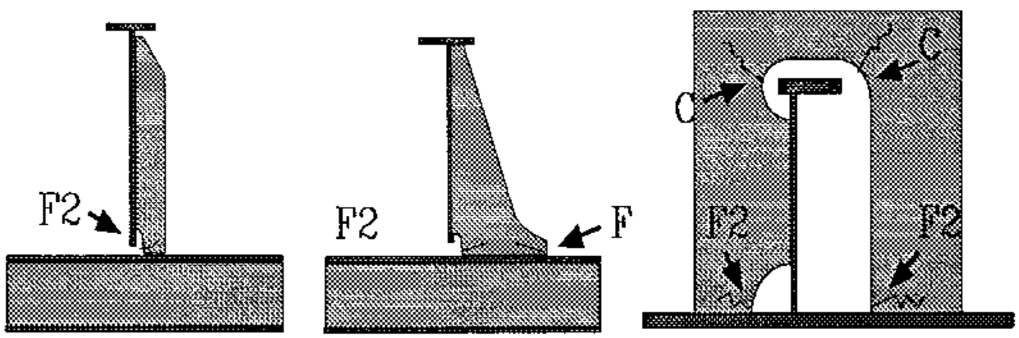
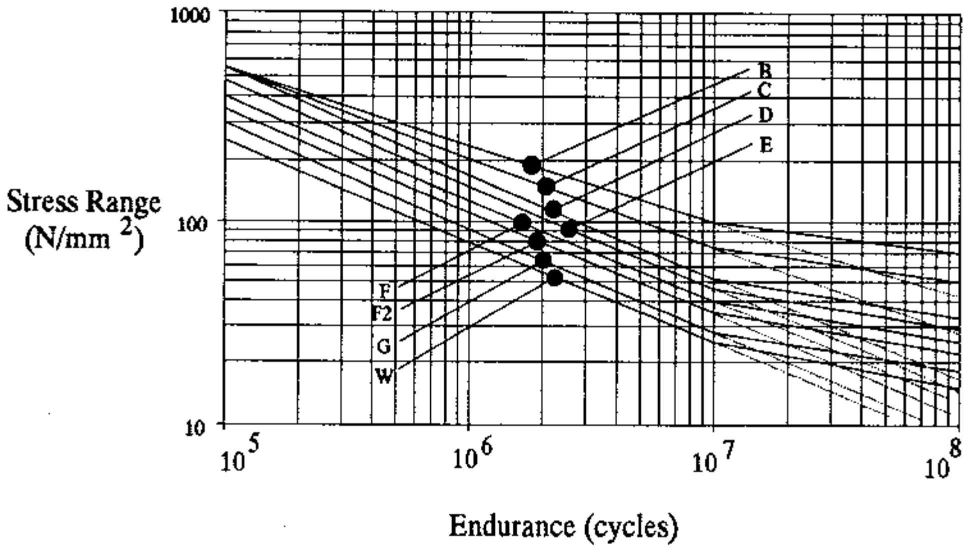


Figure 4.2: S-N class designation on critical structural details



	Parameters $N \leq 10^7$			
Curve Class	A (MPa)	A/A'	m	COV of A*
B	2.34 E15	2.29	4.0	0.44
C	1.08 E14	2.54	3.5	0.50
D	3.99 E12	2.63	3.0	0.51
E	3.29 E12	3.14	3.0	0.63
F	1.73 E12	2.74	3.0	0.54
F2	1.23 E12	2.88	3.0	0.56
G	5.66 E11	2.30	3.0	0.43
W	3.68 E11	2.32	3.0	0.44

Figure 4.3. Mean S-N Curve Constants in Air or Adequately Protected in Seawater [DNV, 1984], [Wirsching, 1987]

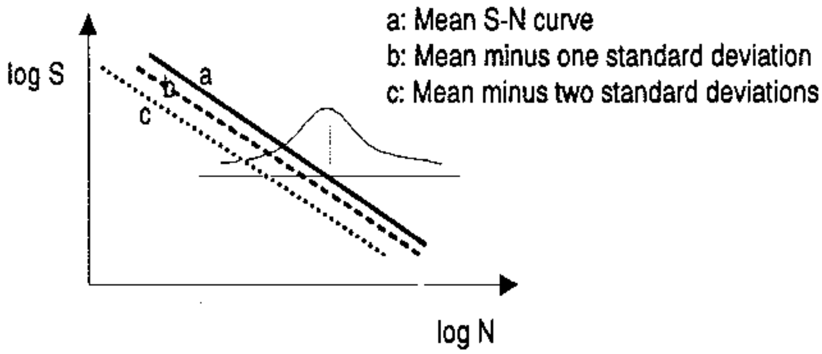


Figure 4.4: S-N curves with different reliability

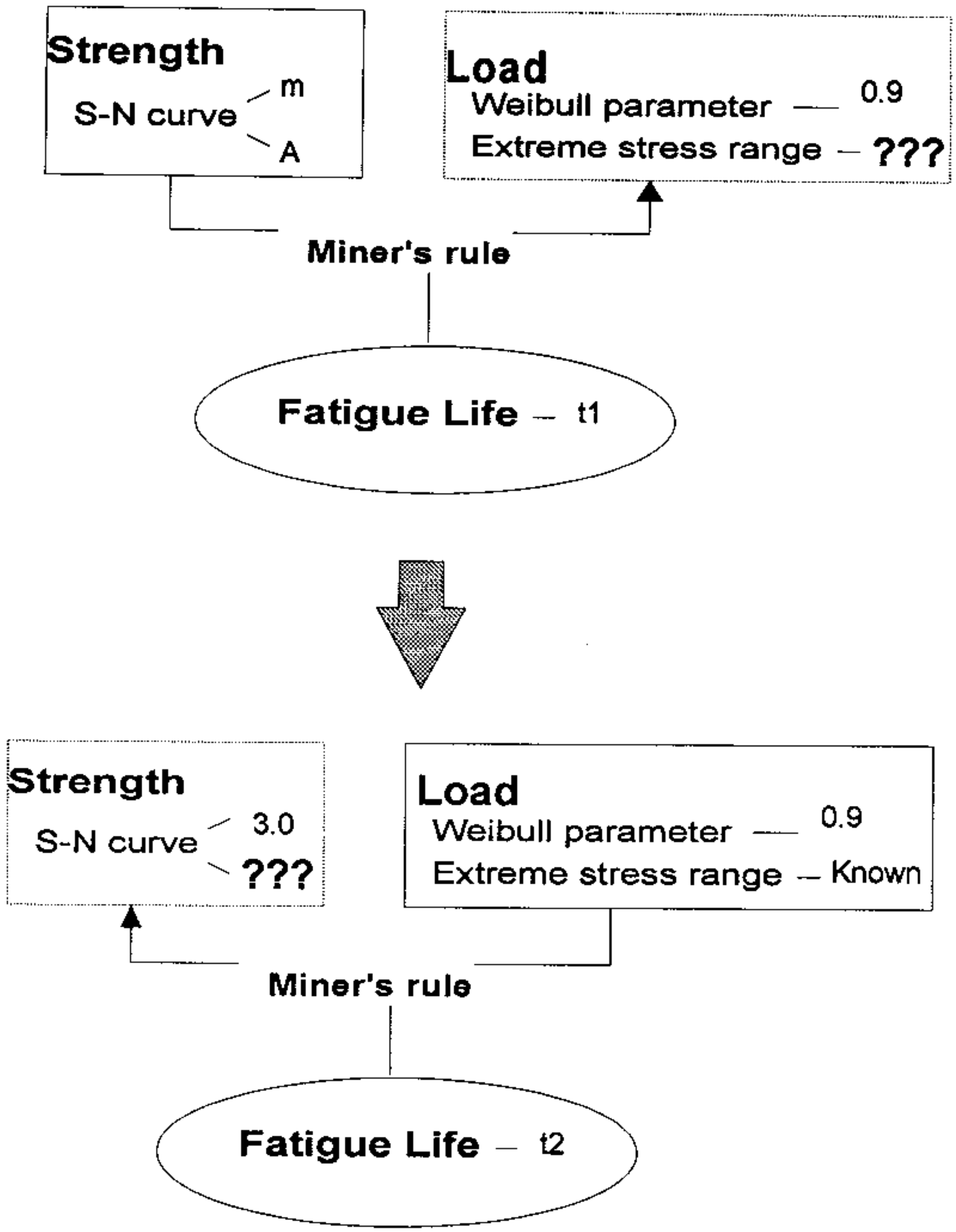


Figure 4.5: S-N calibration process for repaired CSD

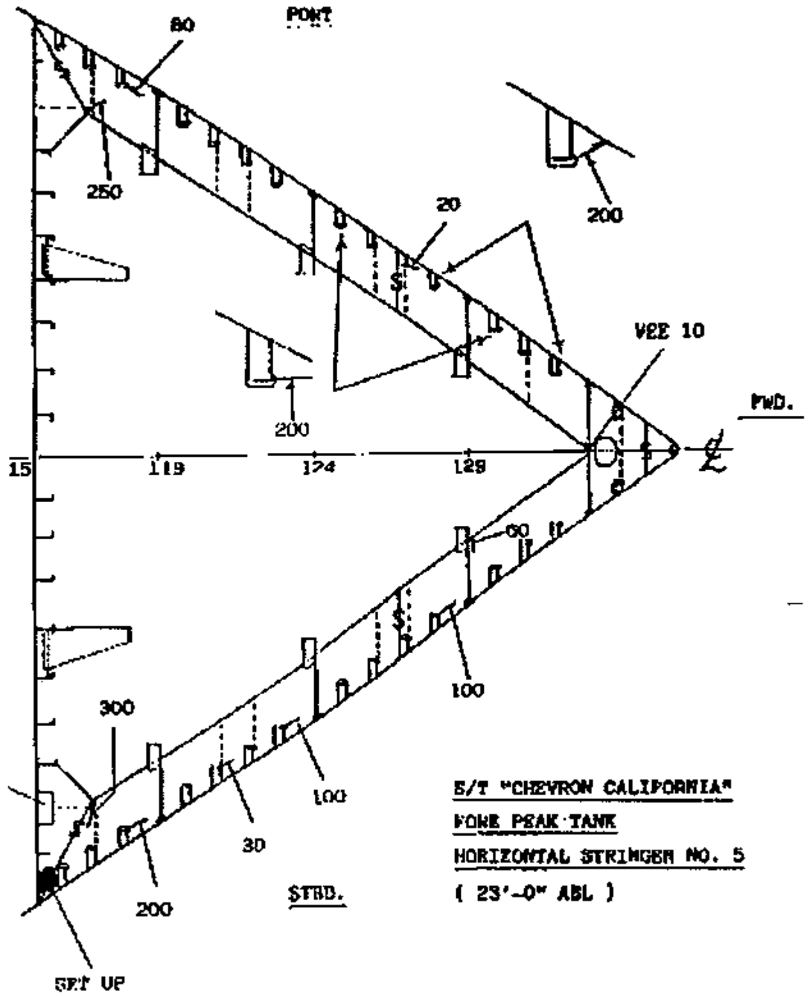


Figure 4.6: Cutout radius failures are shown in the survey report [Chevron, 1987]

264

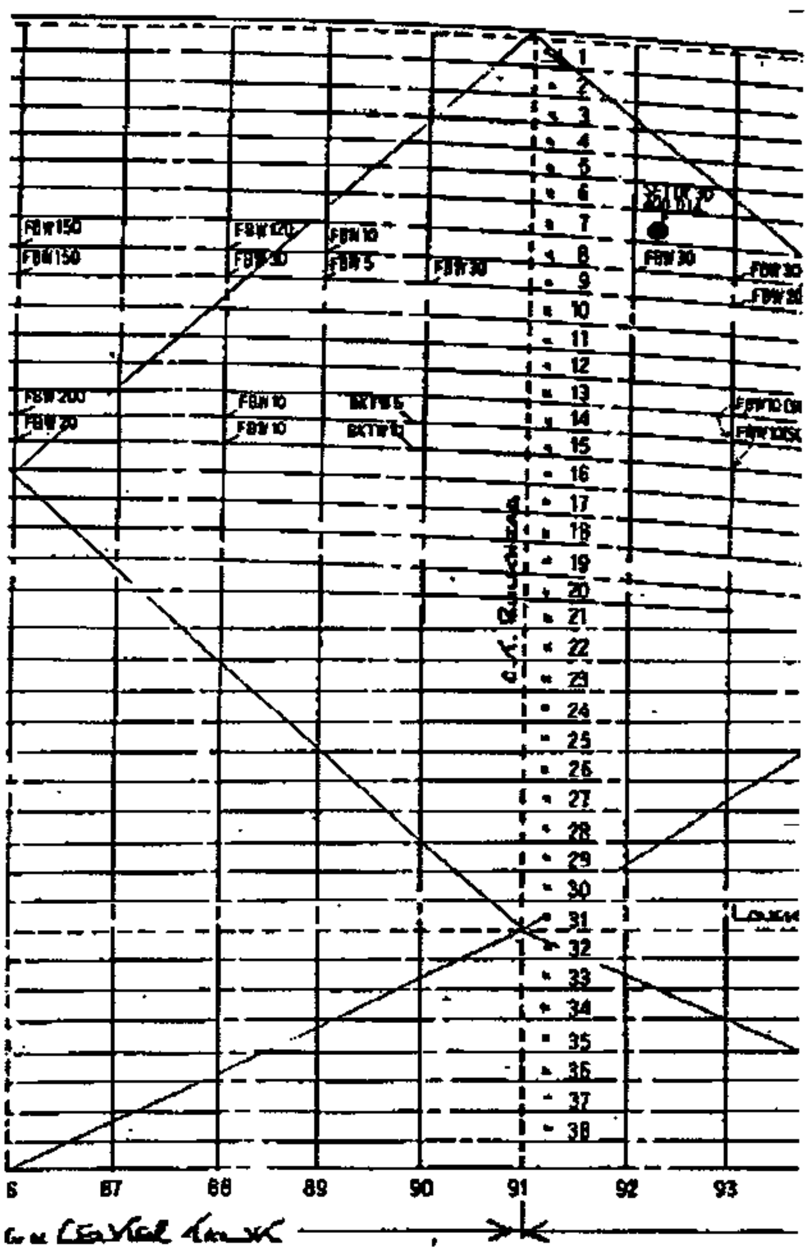


Figure 4.7: Flatbar weld failures are shown in the survey report [Chevron, 1987]

CHAPTER 5. STRESS REDUCTION FACTORS

One of the goals of this study is to establish stress reduction factors (SRFs) for a variety of types of repairs. The objective is to define how much the stress level at a certain location can be changed while reinforcing components are installed. To define these SRFs, finite element analyses (FEA) are performed on some typical Critical Structural Details (CSDs).

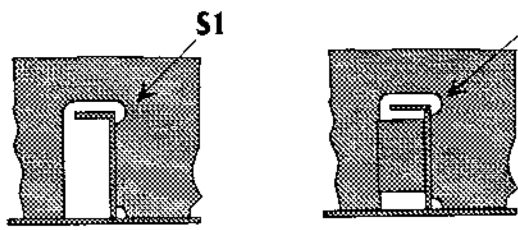
5.1 Determine Stress Reduction Factors by FEA

The stress reduction factor (K_{srf}) is defined as:

$$K_{srf} = \frac{S_2}{S_1} \tag{5.1}$$

where S1 is the hotspot stress before repair.

S2 is the hotspot stress at the same location after a certain repair.



In order to determine one SRF, FEA must be performed on two comparative models (see Figure 5.1 as an example). One FEA is performed and the stress S1 is defined. Then FE model with extra reinforcing components is analyzed to define the stress S2. Then the SRF can be computed. In this chapter, a series of FEA are summarized to determine the SRFs for typical repairs.

A FEA requires a powerful computer with large amount of memory to compute inverse matrixes and a large capacity hard disk to store the results. The computer used in these analyses was an IBM compatible 486-33 PC with a 540 MB hard disk and 20 MB memory. The finite element program utilized was COSMOS/M 1.70. The program is capable of analyzing a model with element number up to 32,000. It contains several modules to deal different types of problems. Only the module of linear static analysis was used in this study.

In order to obtain a hot spot stress, ideally a multi-stage FEA should be performed [Stear and Pauling, 1992]. In this case, a global model is built to represent the overall ship hull structure. Some inter-mediate models are developed to represent a block or a panel of the ship hull. The displacement of the global model can be used as the boundary condition of the inter-mediate models. At last, a detailed local model will be created to represent the complex geometry of the interested area. The hot spot stress can be recovered from the local model. This multi-stage FEA is extremely time consuming.

For RMS, a single-stage analysis is used. In this approach, a single local finite element model is used to capture the response of the critical area to some significant loads. The significant loads includes pressure, longitudinal bending, horizontal bending and shear force. These forces could be estimated by analytical means, and applied to the models directly. Their values depend on where the local model is located. The SRF is therefore dependent on location.

To obtain a SRF at a certain location, loads at the location need to be calculated first so they can be applied to the local model. This makes the process of obtaining SRFs quite tedious. To simplify this process, we assume that the cyclic load that contributes to most of the fatigue damage is cyclic pressure. Therefore only pressure is applied to the local model.

For a linear analysis, the derived stress is proportional to the load (pressure), so the resulting SRFs from either actual pressure or unit pressure will be the same (see Equation 5.2).

$$K_{srf} = \frac{PS_2}{PS_1} = \frac{S_2}{S_1} \tag{5.2}$$

where P is the actual pressure estimated by analytical means. As a result we will apply unit pressure to the CSD models.

The size of the local model must be defined. After researching reports of shell longitudinal-web frame analyses performed by some class societies [DnV, 1991], it was decided to use a model size of one frame bay space by one longitudinal space, with the web frame being modeled up to its face plate (see Figure 5.2). All detail such as brackets, stiffeners and lugs can be included in the model. The model should be large enough so that arbitrary conditions of fixity could be used on the boundaries without risk of grossly effecting the response of the connection.

The conditions of fixity are defined based partially on symmetry and partially on judgment (see Figure 5.2). The aft end of the model was restrained against translation in the longitudinal direction, while both port and starboard sides of the model were restrained against translation in the transverse direction. The reason for applying restraint only on aft end is that the forces due to longitudinal and horizontal bending can be added to the fore end in the future development. The edges of the web were fixed against translation in the vertical direction. Finally both the forward and aft ends of the model were fixed against rotation about the transverse axis.

To simplify the necessary modeling, the connection was slightly altered in that it was assumed that the lug was a butt-welded insert (not lap-welded), and that the flat bar was butt-welded with the top of the longitudinal. Also, no overlap was assumed for the web of the longitudinal past its flange.

The element length (height and width) of the elements near the hot spot should be smaller or equal to the plate thickness according guidelines developed in [Schulte-Strathaus and Bea, 1993]. The accuracy of the finite element results depends on the element size. A finer mesh will in general improve the accuracy of the results, but it will also increase the amount of time necessary for the analysis. For the hot spots in parent material with smooth change of geometry, the exact stress at the hot spot can be calculated by using a fine enough mesh near the hot spot. The calculated hot spot stress converges to the actual stress in the structure with decreasing element size.

Unfortunately for geometrically discontinuous details, e.g. sudden change in the cross section due to a bracket, the hot spot stress does not converge but will keep increasing. Due to the geometric singularity at the hot spots the theoretical stress will reach infinity, which results in the formation of a local plastic zone. A linear elastic FEA can not represent this behavior. A reduction of the element size near the hot spot will therefore result in an increased hot spot stress. In this study, we kept the meshing for two models (with and without repair) consistent, and compared the two hotspot stress to compute the SRF.

5.2 Building FEA Models

There are a variety of designs of longitudinal-web intersections. It is not easy to define a 'typical' CSD. In this project, three CSDs of existing ships to build three sets of longitudinal-web intersection models. These models were analyzed to determine the variation of SRF from one CSD to another. After that, the SRFs were summarized in a way that repair engineers can reference them while making their repair analyses.

All three chosen typical CSDs are longitudinal-transverse intersections on the side shell. Three sets of FEA models have been built based on them. Each set has eight variation models that are created by assembling a lug, a flatbar and/or a backing bracket. A total of twenty-four models were created and analyzed. All models contain one span of transverse web and longitudinal. They are modeled by four node quadrilateral shell elements. Each FEA model consists of 2000 to 4000 elements. A uniform unit pressure load is applied to the shell plate (Figure 5.3).

5.2.1 CSD #1

The scantling of the first CSD is based on the ship drawing of a double-bottom vessel of 190,000 l. tons. The longitudinal-web intersection is located at the side shell near the lower water line. The dimensions are shown in Figure 5.4.

Eight FEA. models were built. The first model is a basic longitudinal-web intersection without a lug, a flatbar or a backing bracket. This model is identified as 'A000'. The letter 'A' stands for Arco. The three zeros stand for no lug (first zero), no flatbar (second zero) and no backing bracket (third zero). Since we are going to investigate the function of the three stiffening components (lug, flatbar and backing bracket), the other seven models were built by adding them to the basic longitudinal-web intersection. The second model is made by adding a lug to the first model. It is identified as 'A100'. Similarly, the rest of the eight models were developed by adding stiffening components. The characteristics of the eight models are summarized as follows:

Model	Lug	Flatbar	Backing Bracket	Equation #	Element #	Node #
A000	No	No	No	28250	4592	4783
A100	Yes	No	No	28856	4694	4884
A010	No	Yes	No	29852	4858	5050
A001	No	No	Yes	28634	4668	4847
A110	Yes	Yes	No	30458	4960	5151

A101	Yes	No	Yes	29240	4770	4948
A011	No	Yes	Yes	30236	4934	5114
A111	Yes	Yes	Yes	30830	5034	5213

5.2.2 CSD #2

The second CSD is based on Chevron's 35,588-ton single skin tanker. Its scantling is shown in Figure 5.5. Eight models were built in the similar way as CSD #1. The characteristics of the eight models are summarized as follows:

Model	Lug	Flatbar	Backing Bracket	Equation #	Element #	Node #
C000	No	No	No	21211	3441	3602
C100	Yes	No	No	21451	3486	3642
C010	No	Yes	No	22657	3691	3843
C001	No	No	Yes	21595	3510	3666
C110	Yes	Yes	No	22897	3736	3883
C101	Yes	No	Yes	21835	3555	3706
C011	No	Yes	Yes	23041	3760	3907
C111	Yes	Yes	Yes	23281	3805	3947

5.2.3 CSD #3

The last CSD is based on a double hull tanker that is being building by Newport News. Its scantling is shown in Figure 5.6. Eight models were built in the similar way as CSD #1 and CSD #2. The characteristics of the eight models are summarized as follows:

Model	Lug	Flatbar	Backing Bracket	Equation #	Element #	Node #
N000	No	No	No	17673	2888	3017
N100	Yes	No	No	17835	2920	3044
N010	No	Yes	No	19093	3128	3257
N001	No	No	Yes	18057	2956	3081

N110	Yes	Yes	No	19255	3160	3284
N101	Yes	No	Yes	18219	2988	3108
N011	No	Yes	Yes	19477	3196	3321
N111	Yes	Yes	Yes	19639	3228	3348

5.3 Establishing Stress Reduction Factors

From the results of the FEA, it was found that the three CSDs have very similar stress patterns, although their scantlings and geometry are different (Figure 5.7, 5.8 and 5.9). The first models of all three CSDs are basic longitudinal-web intersections; they do not have any repair components. This model is not common in ships because most would have been stiffened by a flatbar. However a very similar CSD exists at the intersection of horizontal stringer and transverse stiffener in the forepeak tanks of some ships. The results of the FEA shows that this kind of geometry has a stress concentration in the cutout radius for all three CSDs (see Figure 5.7, 5.8 and 5.9). The FEA results agree well with our experience. A tanker survey report documents a large number of cracks in the cutout radius of this type of detail [Chevron, 1982; 1984].

In order to overcome the stress concentration in the cutout radius, one component is added to the basic CSD to reduce the stress. First, a lug was added. The models (A100, C100 and N100) were developed (see Appendix B). For model A, the stress after adding a lug is reduced to 70.0% of its original stress value. For model C, the ratio is 83.0%. For model N, the ratio is 75.2%. It can be concluded that based on the three CSDs the SRF is 76.1% on average for adding a lug to a basic longitudinal-web intersection (see Figure 5.10).

Instead of adding a lug, one could try adding a flatbar to reduce the high stress in the cutout radius. By comparing the stresses between two sets of models (X000 and X010), it was found that the SRFs are 65.9%, 46.4% and 55.8% for model A, C, and N, respectively. Therefore the average SRF for adding a flatbar is 56.0%. This SRF looks more attractive than the one of adding a lug (76.1%). However, while carefully examining the stress contours of model X010, it was found that adding a flatbar created a new hotspot in the flatbar heel. For model A, the new hotspot stress is higher than the cutout-radius stress in the model without repair. This means that while a problem is solved in the cutout radius, a more serious problem is created in the flatbar heel. Careful use of the SRFs is important.

The third option of resolving the problem in cutout radius is to add a bracket. This option has a averaged SRF of 92.2%. Also, it has a very similar result to the one of adding a flatbar. Apparently, this is the worst repair alternative.

Three more models (X110, X011 and X101) were made by adding two components to the basic model. The first model in this category (X110) was made by adding a lug and a flatbar. The stresses in the cutout radius of these models were reduced significantly (see Figure 5.7, 5.8, 5.9). However, the flatbar heel suffers the same high stresses as in the model with a flatbar. The third model X101 has stress contours similar to the first model X110. The second model (X011) produced a very good stress contour where the cutout radius, bracket heel, and flatbar heel all have very low stresses. The high stress locations were moved to flatbar and bracket toes. While reducing the high stress by applying soft toe to the flatbar and bracket, this model may be a very good design.

The last model (X111) in the eight models has three reinforcing components (lug, flatbar and bracket). With a lug that model X011 does not include, it has a stress contour similar to model X011 (see Appendix B). Therefore it may not be worthwhile to use a lug, if the CSD has both a flatbar and a bracket. This model is definitely most robust among the eight, but it may not be the most cost effective option.

By organizing and comparing the results of FEA., a summary has been developed for the SRFs of different repair alternatives (see Figure 5.10). Four common CSDs in ships are listed with possible crack locations, possible repair alternatives and their SRFs. When a repair analysis is intended but without available time for FEA., the SRFs can be applied to make a quick estimate of the fatigue life.

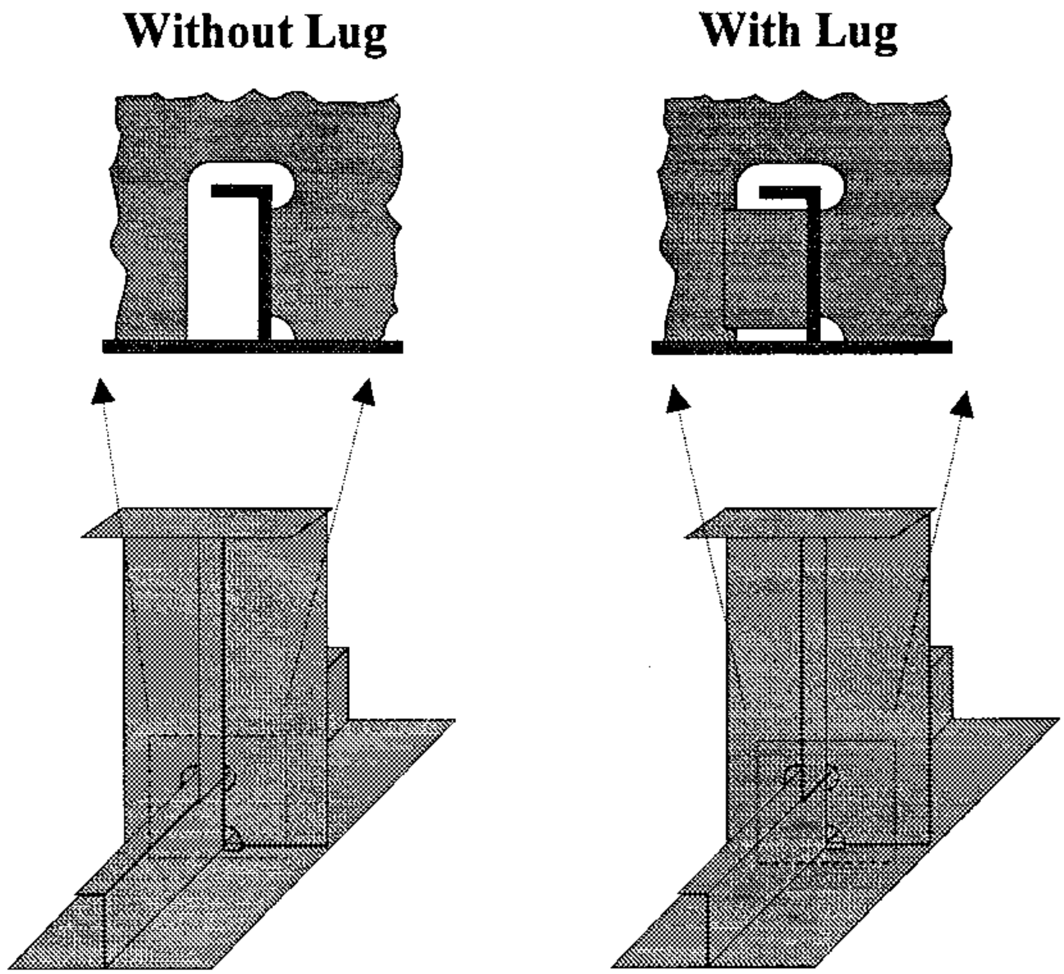


Figure 5.1: Models with and without a lug

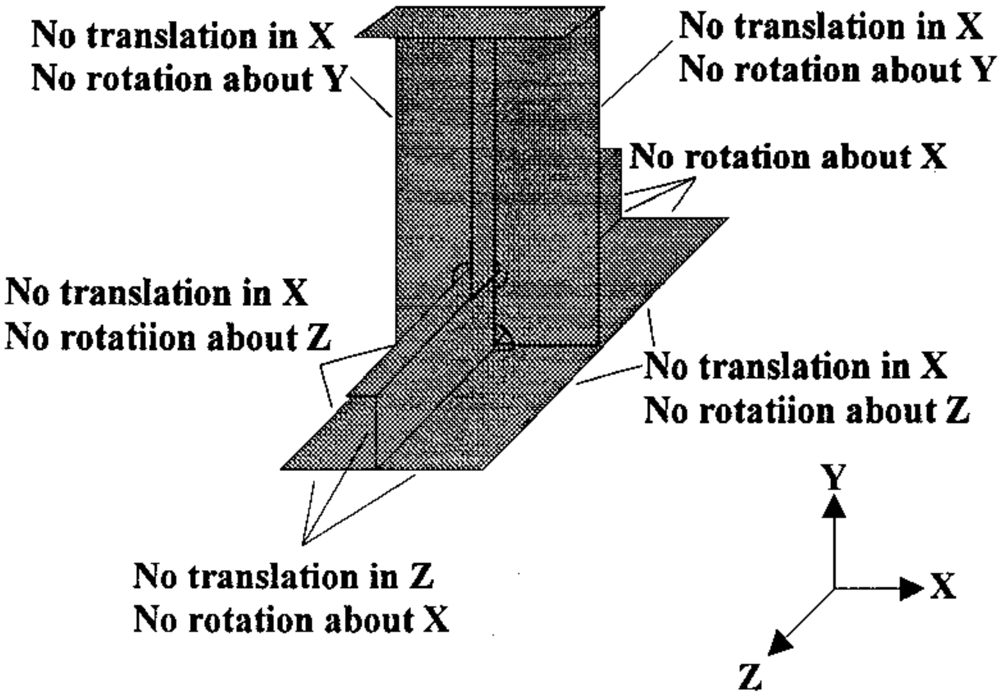


Figure 5.2: Boundary condition

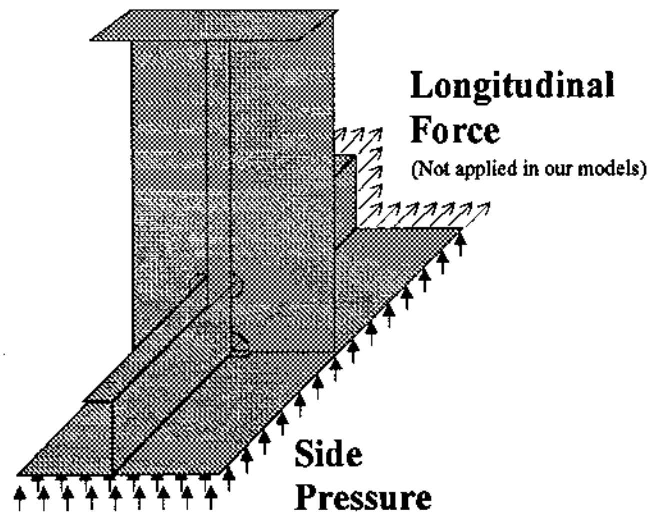


Figure 5.3: Load condition

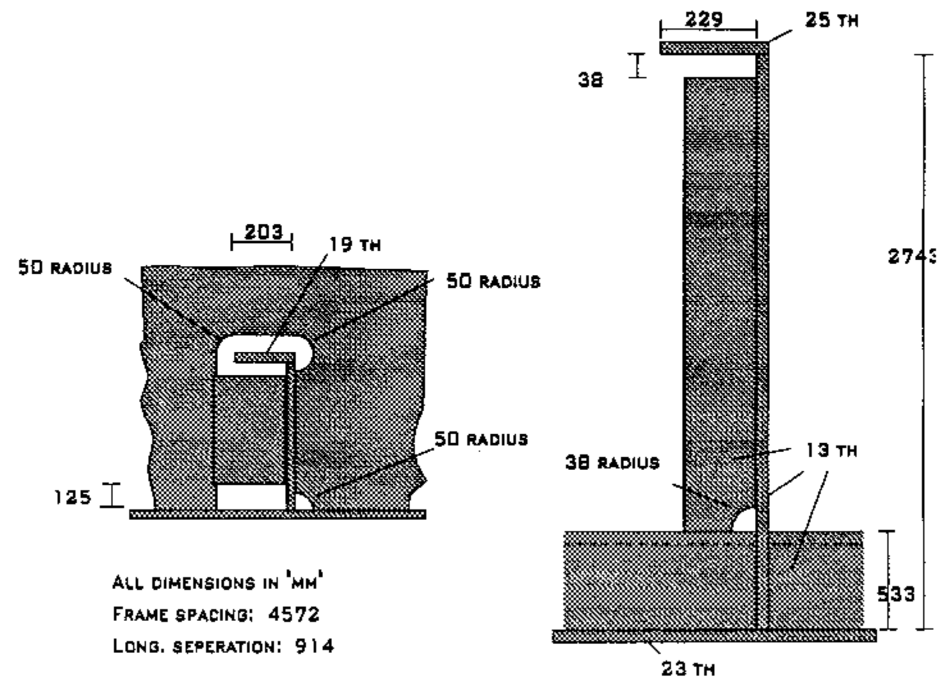


Figure 5.4: Scantling of CSD #1 (for model Axxx)

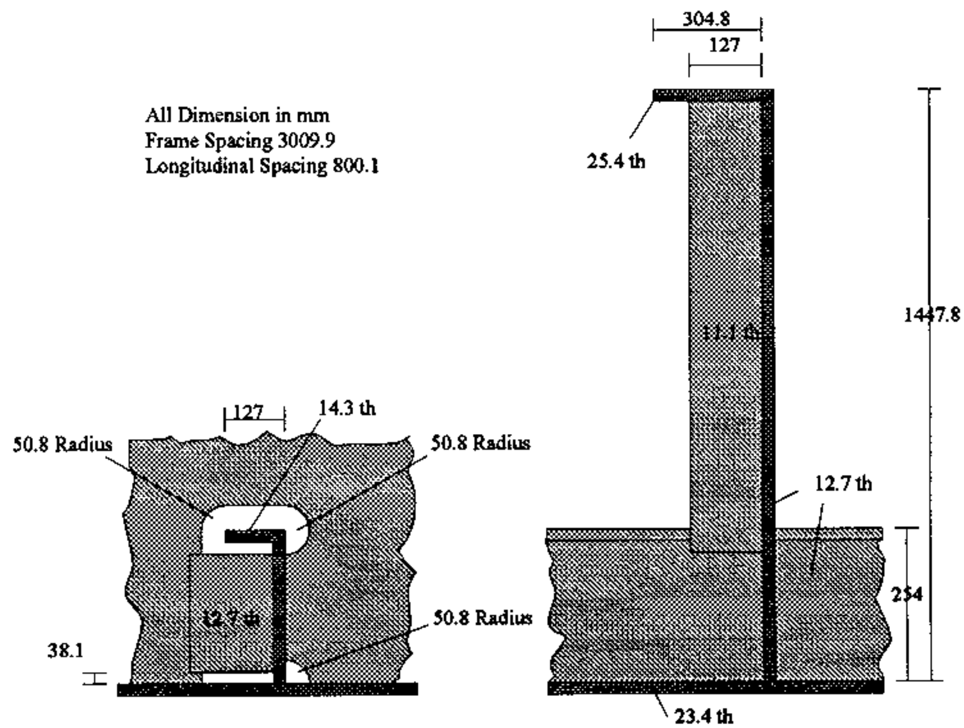


Figure 5.5: Scantling of CSD #2 (for model Cxxx)

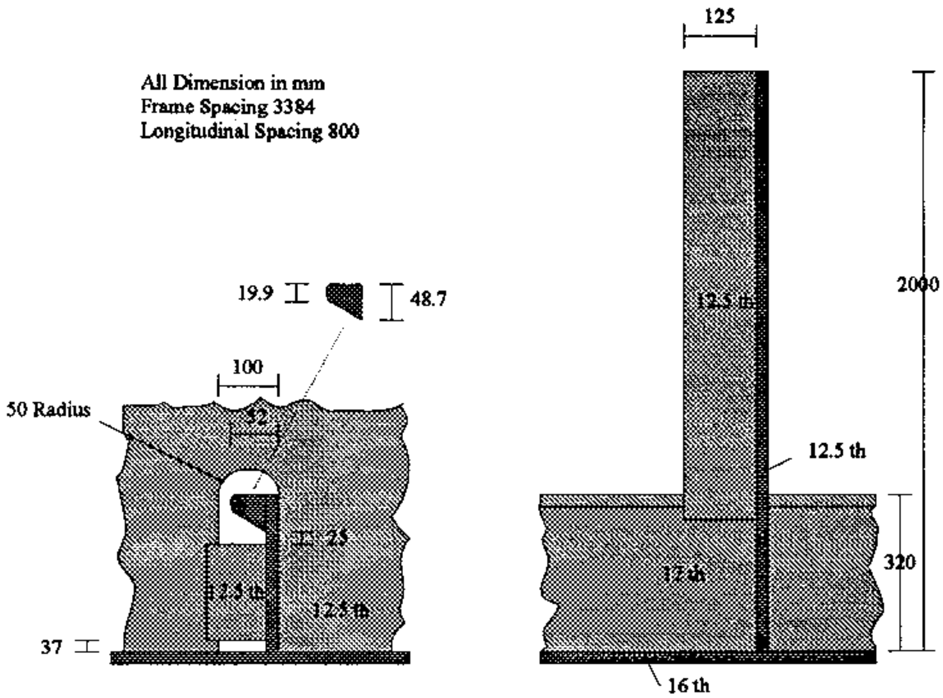


Figure 5.6: Scantling of CSD #3 (for model Nxxx)

CSD "A"

Stresses in CSD "A" which is under unit pressure load on its shell plate.

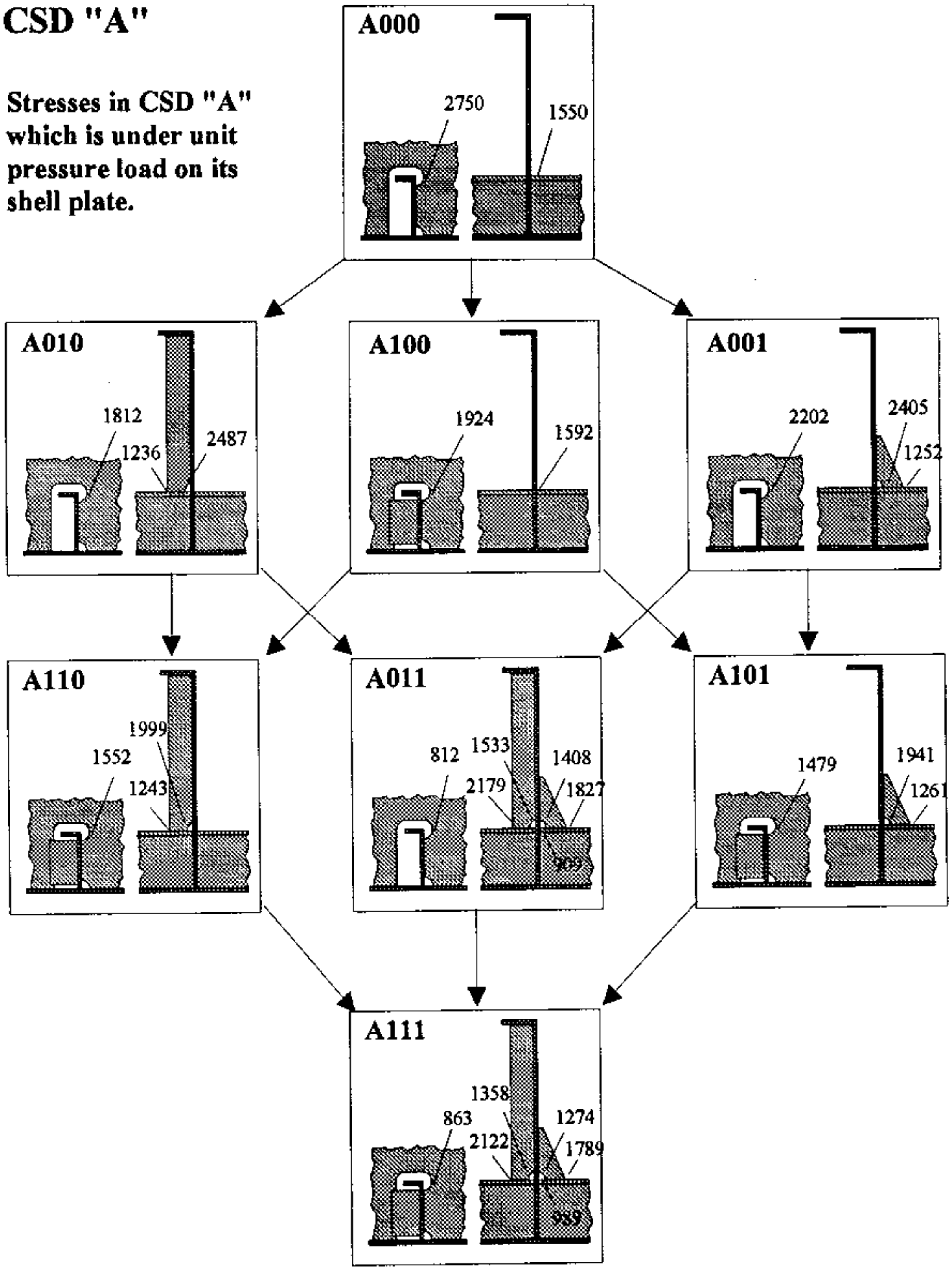


Figure 5.7: Result of stresses in CSD "A" from FEA.

CSD "C"

Stresses in CSD "C" which is under unit pressure load on its shell plate.

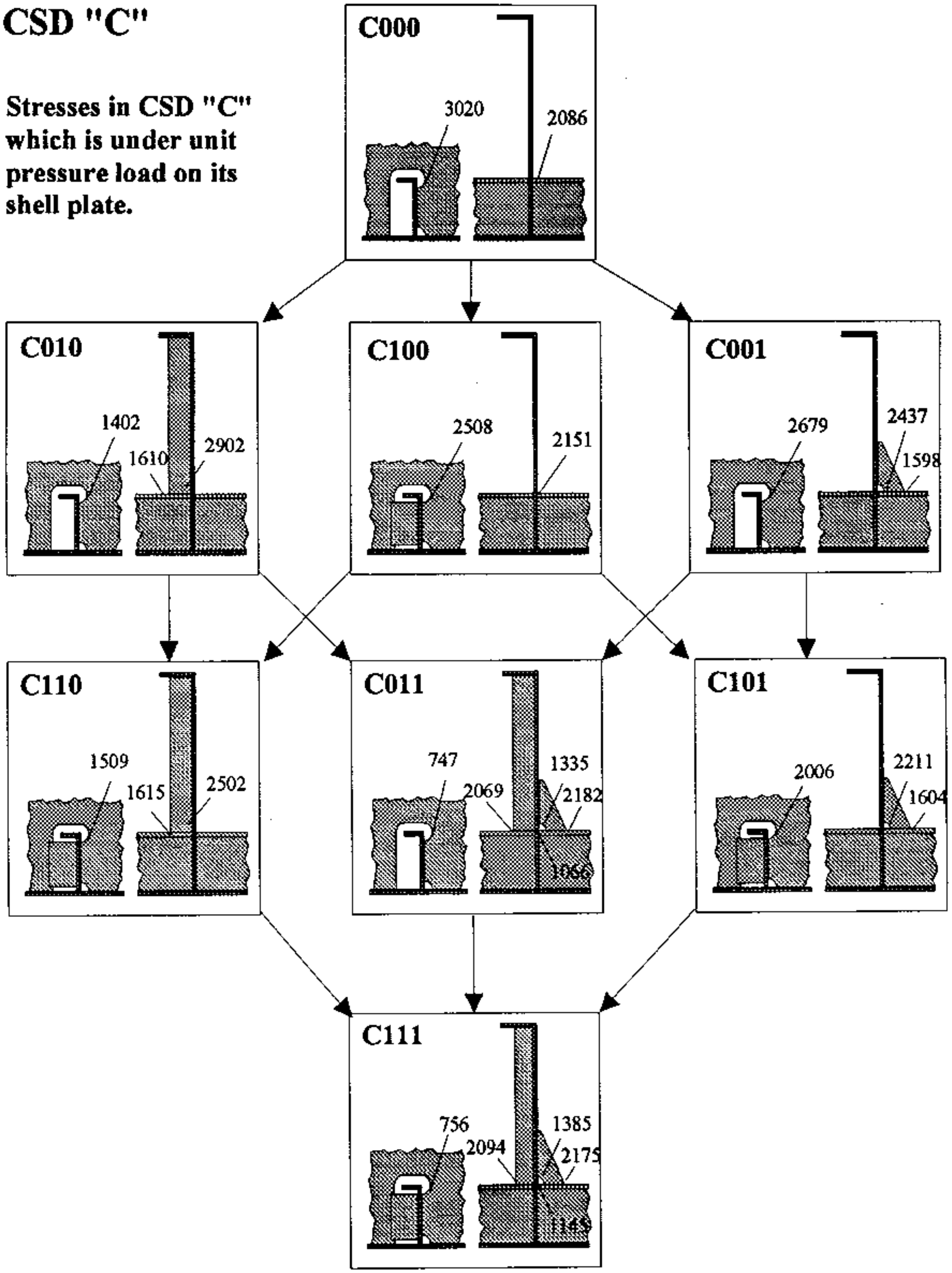


Figure 5.8: Result of stresses in CSD "C" from FEA.

CSD "N"

Stresses in CSD "N" which is under unit pressure load on its shell plate.

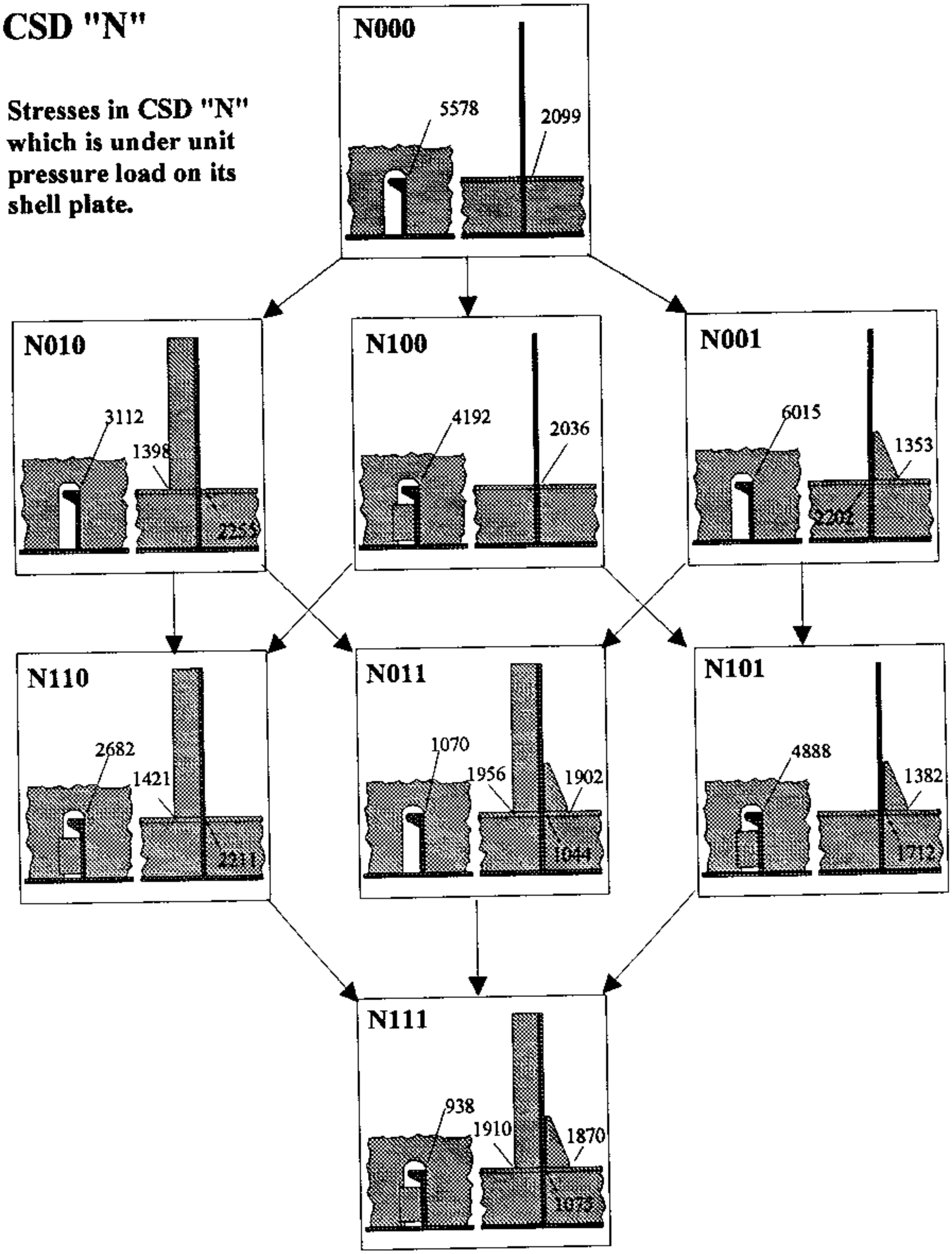


Figure 5.9: Result of stresses in CSD "N" from FEA.

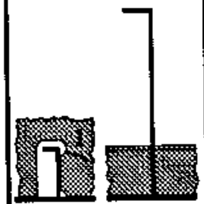
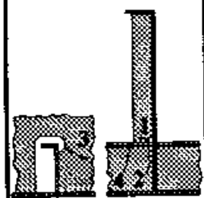
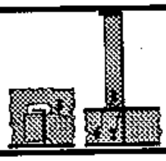
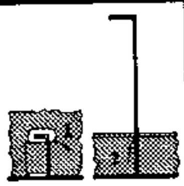
Stress Reduction Factors (SRF)								
If your fatigue crack is like:		Repair by adding:	SRF (%) A	C	N	Avg.	New hot spot at: (%)	
	1	Lug	70.0	83.0	75.2	**76.1**	F.B. heel	93.3
		Flatbar	65.9	46.4	55.8	**56.0**		
		Bracket	80.0	88.7	107.8	**92.2**	Brac. heel	84.1
		Flatbar & Lug	56.4	50.0	48.1	**51.5**	F.B. heel	77.8
		Flatbar & Bracket	30.0	24.7	19.2	**24.6**	F.B. toe	61.2
		Bracket & Lug	53.8	66.4	87.6	**69.3**	Brac. heel	71.9
		Bracket, Lug & F.	31.4	25.0	16.8	**24.4**	F.B. toe	60.2
	1	Lug	80.4	82.9	N/A	**81.7**		
		Bracket	61.6	46.0	N/A	**53.8**	F.B. toe	80.0
		Bracket & Lug	54.6	47.7	N/A	**51.2**	F.B. toe	78.8
	2	Lug	93.9	98.1	98.0	**96.7**		
		Bracket	47.0	41.8	46.3	**45.0**	F.B. toe	94.4
		Bracket & Lug	51.2	44.9	47.7	**47.9**	F.B. toe	92.7
	3	Lug	85.7	107.6	86.2	**93.2**		
		Bracket	44.8	53.3	34.4	**44.2**	F.B. toe	111.0
		Bracket & Lug	47.7	53.9	30.1	**43.9**	F.B. toe	109.3
	4	Lug	100.6	100.4	101.6	**100.9**		
		Bracket	143.7	130.5	124.7	**133.0**		
		Bracket & Lug	142.1	130.1	124.2	**132.1**		
	1	Bracket	68.0	57.5	N/A	**62.8**	F.B. toe	96.6
	2	Bracket	54.4	45.8	48.6	**49.6**	F.B. toe	96.2
	3	Bracket	55.6	50.1	35.0	**46.9**	F.B. toe	115.6
	4	Bracket	141.3	129.7	122.2	**131.1**		
	1	Flatbar	80.6	60.2	64.0	**68.3**	F.B. heel	101.9
		Bracket	76.9	80.0	116.6	**91.2**	Brac. heel	94.6
		Flatbar & Bracket	44.9	30.1	22.4	**32.5**	F.B. toe	79.8
	2	Flatbar	112.5	116.3	108.6	**112.5**	F.B. heel	118.5
		Bracket	114.1	102.8	84.3	**100.4**	Brac. heel	89.5
		Flatbar & Bracket	62.1	53.2	52.8	**56.0**	F.B. toe	108.2

Figure 5.10: Summary of Stress Reduction Factors for typical fatigue repairs

CHAPTER 6. FATIGUE CRACK REPAIR PROGRAM

One of the goals of the RMS study was to develop and verify an analytical tool for repair evaluations. This chapter reviews a computer program that demonstrates the concept and feasibility of RMS. The procedures of simplified fatigue analysis have been written into subroutines (functions) using programming language C.

According to results from questionnaires sent out to industrial organizations in the early stage of this research [Gallion, 1992], a graphical database of repaired alternatives was a favorable feature to be associated in the PC software. Therefore the computer program is developed for the environment of MS Windows to take the advantage of a graphical interface.

In this chapter, a summary of the program is presented. A verification of the program follows.

6.1 RMS Program

The source code directory contains the following files:

Repair.prj	Project file.
Repair.def	Define program environment.
Repair.res	Supplies bitmaps, menu, dialogue boxes, cursor and other resources.
Repair.h	Define public structures and associated constants.
Main.c	Does initialization and created the Main Window.
MainWnd.c	Processes window messages.
FileCmds.c	Performs File Commands for the top Menu Bar.
FileFmt.c	Writes the different file formats.

FileUtil.c	Provides common procedures for file commands.

Analysis.c	Shows data input windows for fatigue life prediction.

CalcuFat.c	Calculates fatigue life.

AddRecrd.c	Processes crack record input dialogue.

Figure 6.1 shows the calling relationship between the functions.

Due to the large amount of source code, only one module of the source code is listed in Appendix A. The source code for the remaining modules is documented in a separate report [Ma & Bea, 1993]. The purposes of the source code are briefed below in terms of each RMS module. The basic structure of the code is shown in Figure 6.1.

6.1.1 Windows Module

This module includes Main.c and MainWnd.c. It does initialization and created the main window with a menu bar. It processes window messages like mouse moving, resizing windows, user selecting a command, inputting form keyboard and others.

6.1.2 File Input Output Module

This module includes FileCmds.c, FileFmt.c, and FileUtil.c. These files provide functions that can input a text file like *.rms and also there are functions for importing bitmap files.

6.1.3 Crack Management Database Module

This module mainly includes AddRecrd.c. It let users to add, delete or edit a record for a particular crack in the graphical ship layout.

6.1.4 Failure Diagnosis Module

No failure diagnosis is conducted. The program assumes the mode of failure is fatigue and the cause of failure is not due to poor quality control at initial construction or due to corrosive effects.

6.1.5 Repair Alternatives Selection (Analysis) Module

This module is the code file, Analysis.c. Detail configurations for any component group (e.g., side shell components) are built into a dialogue box in the program. The graphical detail-type-selection dialogue box allows users to select different detail types (e.g., longitudinal cutout, flatbar, bracket) and the modified design of each structural detail. When redesigning the detail, the original crack location may be either welded or replaced. Since the mode of failure is fatigue, only the crack repair options are considered. These options include vee and weld, vee and weld plus post-weld improvement, add insert plate, and redesign of the detail. The desired repair option can then be selected by the user. In the case of redesign, the user selects from a list of valid detail configurations.

6.1.6 Fatigue Analysis Module

This module is basically the code file, CalcuFat.c. The necessary information to conduct the repair analysis is provided by interactive input from the user and pre-defined data in the program. Ship loading information, including the Weibull parameter, average stress frequency, and expert load zones and ratios is pre-defined in the program. Stress reduction factors for each configuration, and S-N class designations for each location are pre-defined, too. Interactive input includes detail configuration and failure location, the mean time to failure of the original detail and the desired repair option. With all the information above, the program calculates the expected repair life by using the approach developed in Chapter 3. Repair analysis is conducted only at the location of failure.

6.1.7 Help Module

This module is in the code file, MainWnd.c, along with help script files, Repair1.hlp and HelpHow.hlp. It performs the commands 'How to Use RMS' and 'Repair Information' under the menu bar in the window. The former instructs users how to use all the command in the RMS window. The latter provide users a general introduction on ship maintenance and repair.

Due to the limited time available, RMS program has been developed into a prototype that provide only necessary functions. For a more powerful application, the RMS may need more coding effort to enhance the current version of RMS.

6.2 Demonstration and Verification

To demonstrate and verify the code, the RMS is applied to a case study of a failure in a side shell structure. The repair of a crack in the longitudinal cutout shown in Figure 6.2 is explored. Assume that this crack is found when the ship is 10 years old. This means the time needed for this critical location to crack and develop to the current particular length is about 10 years. The ship owner plans to operate this ship for another

15 years, and would like to define the alternative types of repair that are available and determine which one could survive for 15 more years without re-cracking.

The solution to the ship owner's question will be developed with the RMS program. The following paragraphs illustrate the application of the RMS program.

First, activate the RMS program. Select Repair-Analysis under the menu bar, and choose the cracked detail as in Figure 6.3. Another dialogue box will pop up to let you enter the failure time (Figure 6.4). Since the crack takes 10 years to grow up, enter the number '10'.

At this point several possible repair alternatives will be introduced. A simplified fatigue analysis will be carried out for each of them (Figure 6.5).

The results (Figure 6.5) show that the vee and weld can last about only 4 years. These results match the experience of ship structural repairs quite well. Repair by veeing and welding usually fails again very soon. Apparently this repair is not robust enough to survive the rest 10 years of the ship life in this example.

The second repair 'Weld Plus Post-weld Improvement' will last about 4.7 years. The third repair, inserting a new plate, is something like re-running the fatigue damage cumulation from the starting point of the structure life. It is reasonable to take another 10 years to re-crack and grow to the same length. This repair may not provide sufficient repair life.

The rest of the repairs are made by adding a lug. They have repair lives better than those without adding a lug. Therefore the better repairs in this case would be repair #3 or #6.

Note the stress reduction factors in the S-N data file 'REPAIR.DAT' is defined based on finite element analysis (see Chapter 5). To draw more conclusions from this case study, a review of the relative costs, expected interest rates, and the load ratios is necessary. All these will have a significant impact on the decision. With a large database of available repair options, a complete version of the RMS would be a valuable tool for the assessment of repair options.

6.3 RMS Crack Database System

6.3.1 Database Considerations

Through a ship's life, a number of surveys will be carried out. Thousands of pieces of data on coatings, fractures, and gaugings will be recorded in each survey. Due to the amount of survey data, the data are usually difficult and expensive to record, retrieve and analyze. The data can consist of rough sketches in a repair superintendent's notebook and shipyard invoices collected in a repair file. Data that resides in the experience of individuals involved in ship maintenance also needs to be archived.

The gathering, storage, retrieval, and analysis of the huge quantity of the information can be facilitated by developing a computer database system [Ma and Bea, 1992]. Database systems can significantly improve the efficiency and effectiveness of ship maintenance. Development of maintenance plans, specifications, and reports can be greatly facilitated with the help of such systems. In general, database systems are not well developed in the ship industry compared with those of other industries. Some industrial organizations have pioneered the development of computer based database systems for ship structures. At the present time, these systems are still in their early stages of development. The RMS program includes a simple but powerful graphical crack database.

The general objectives of an RMS database system development are as follows:

- Collect meaningful data.
- Store the data.

- Provide means for logical data management.
- Provide access to the relevant data easily.
- Allow for the organization of the data in a form suitable for analyses.
- Analyze the data.
- Show trends of the data.
- Communicate and report the data.

Some of these objectives have been fulfilled in the RMS program. It is feasible to fulfill all of them. However, it will take more time and effort to complete the RMS.

Figure 6.6 shows the basic parts of a RMS flow chart for inspection, maintenance and repair. Once a ship is ready for service, a series of surveys can be scheduled according the inspection program. The objective and scope of the internal structural inspections are defined. The access methods and data recording methods are chosen, and then the survey is performed. The survey results including corrosion gaugings, fatigue cracks, status of coating and corrosion protection system, or other structural defects are updated into the corresponding databases. Using the survey data, a Repair Management System evaluates repair alternatives. Finally the repairs are carried out.

The overall advantage of such a comprehensive database system is that the data are in electronic format so that the data can be transferred more easily and faster by modems or floppy diskettes. The data can be transmitted among ship owners, shipyards, repair yards, and design offices via telephone and satellite communication. It also can enhance the efficiency of inspection, maintenance, and repair by eliminating manual writing of the steel repair specification or manual drafting of repair drawings. In addition, it provides the capacity to quickly update corrosion, fatigue, and repair databases.

6.3.2 RMS Crack Database

A crack database has been developed in the RMS program. The RMS crack database can handle only one failure mode, fatigue cracking. It stores the general information on a ship including a ship three-view layout, three classes of crack on a ship layout, a crack detail information and the drawing of cracked structural details. Since it is a prototype of a graphical database, it can be further developed into a powerful database with ability to handle corrosion gaugings, fatigue crack, coating status, and other items of information needed by ship maintenance.

This database has three pre-defined ship layouts including a single hull tanker, a bulkcarrier and a container ship (see Figure 6.7). The three layouts use most typical configurations such that most user can simply adopt them as their ship drawings. It also provides a user input mode to allow users import their own ship drawings. By this way, all ship types can utilize the functions of RMS as long as they have their ship drawings. The ship drawing can be scanned into a bitmap file easily or users can draw them by using some MS-Windows based drawing programs. The general information of the ship can be stored in 8 input fields: ship name, vessel class, owner name, classification, builder name, delivery date, service route and additional information (Figure 6.9). Each of these fields can store 25 characters.

The RMS program uses three-view ship layouts to locate a crack. This means that users have to input three crack-marks to identify one single crack. The three-view ship layout successfully solves the difficulty of describing the spatial location within a hull structure of a particular survey result (See Figure 6.8). The graphics tell users the coordinate system clearly. And the precise location can be emphasized again by the character based crack data that is input by users.

According to U.S. Coast Guard's regulations [USCG, 1990], ship structural failure can be classified into three classes depending on the size of failure and the potential danger. Therefore the RMS program uses different colors and sizes of crack mark to distinguish the classes. A large red star sign is assigned to indicate a Class 1 crack. A blue star sign is for Class 2, and a green one for Class 3. The definitions of the three classes are summarized as follows:

- ***Class 1 Structural Failure:*** *During normal operating conditions, either (1) a fracture of the oil/watertight envelope that is visible and of any length, or a buckle, that has either initiated in or has propagated into the oil/watertight envelope of a vessel, or (2) a fracture 10 feet or longer in length that has either initiated in or has propagated into an internal strength member.*
- ***Class 2 Structural Failure:*** *A fracture less than 10 feet in length, or a buckle, that has either initiated in or has propagated into an internal strength member during normal operating conditions.*
- ***Class 3 Structural Failure:*** *A fracture or buckle that occur under normal operating conditions that does not otherwise meet the definition of either a Class 1 or Class 2 structural failure.*

For each crack there are five fields to be input: crack location, finding date, length, repair status and comments. All the fields are character based, so users do not need to memorize any keywords and can simply type in text. The location field can hold 24 characters, and the finding date for 12, the length for 10, the repair status for 25 and the comments for 25 characters. Users can also attach a graphic of a corresponding cracked structural detail to the crack data (Figure 6.10 and 6.11). A library of 13 cracked structural details has been created to help users select a graphic easily (Figure

6.12). A user input mode is also provided to allow users import their own structural detail drawings.

In summary, the RMS crack database has the following advantages:

1. It is a user friendly database. All the operation follows the standards of the MS-Windows environment.
2. Inspectors can print out the ship layout that is pre-defined in the RMS as a draft paper before going into tanks. Also, they can review the previous data of RMS database to locate the critical area with high likelihood of failures. The RMS simplifies the final inspection report. The final report consists of a floppy disk containing the crack information and the comments regarding the vessel inspection.
4. The RMS has the ability to analyze and evaluate the best repair from a group of repair alternatives.
5. It enhances the efficiency and quality of the inspection and repair. The inspection team and the repair team can both communicate with the home office naval architect, transmitting copies of the information contained on the floppy disks via satellite communications. Naval architects in the home office can then participate in decisions to modify the inspection program or to change the repair specification.
6. The RMS uses a three-view ship layout, a character based description of crack location and a library of cracked detail drawings to specify a particular crack. It is easy to use and understand. Most importantly, it is easy to create a ship layout or a structural detail drawing. Some other database systems that use CAD to locate a crack or other failures can be difficult to operate. In addition, creating a CAD ship model takes a lot of effort, time and money. Other databases that use

keywords to specify a failure location without graphical operating environment are difficult to use.

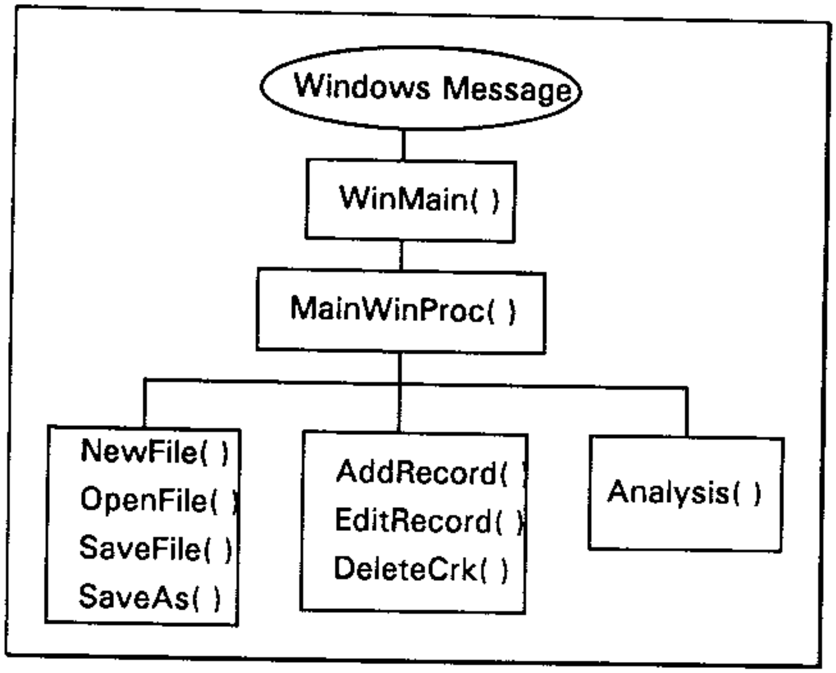


Figure 6.1: This shows the message flow of the RMS program.

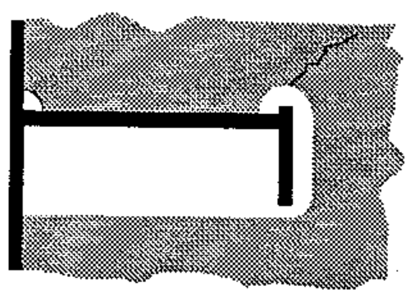


Figure 6.2: A crack is found around longitudinal cutout.

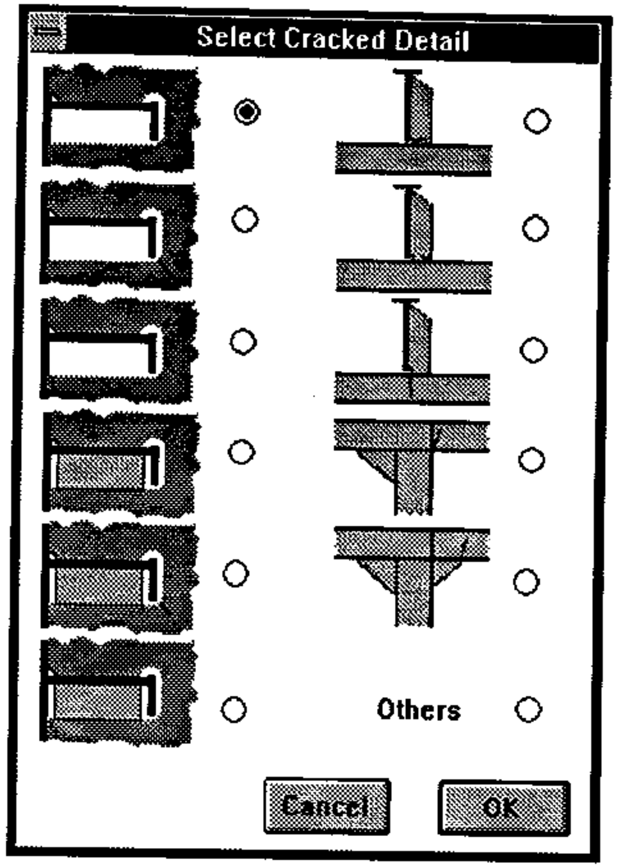


Figure 6.3: Specify the crack location.

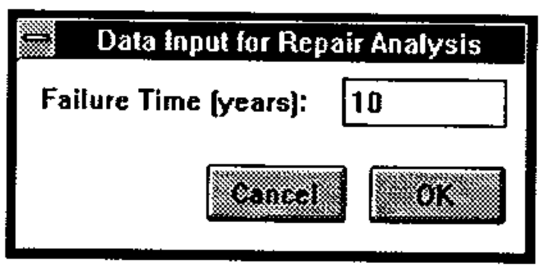


Figure 6.4: Input failure time.

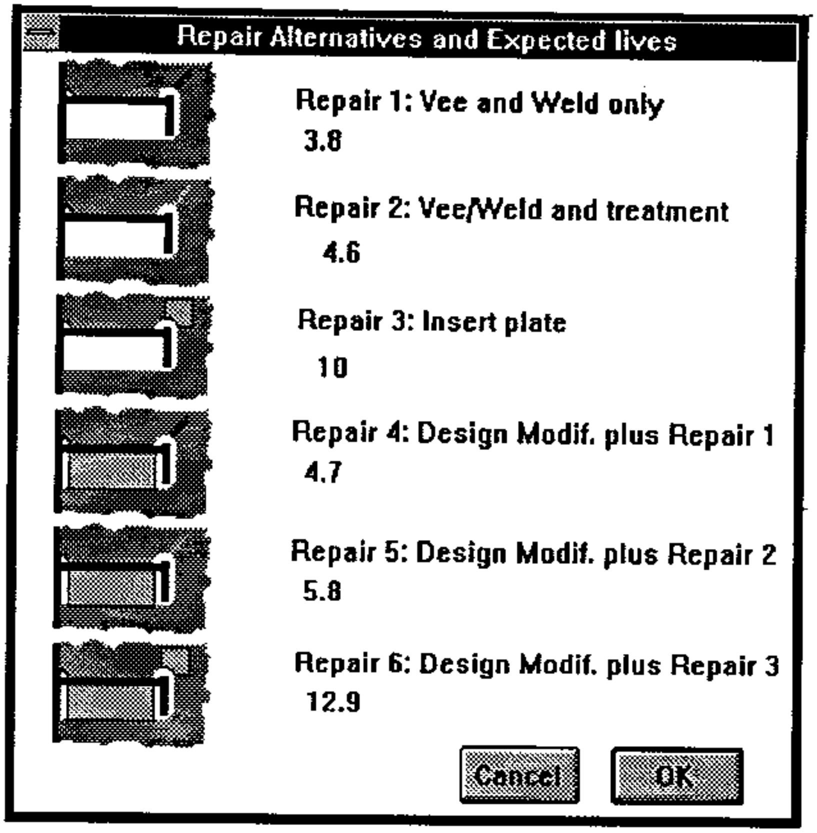


Figure 6.5: The RMS program shows the result of the expected fatigue lives.

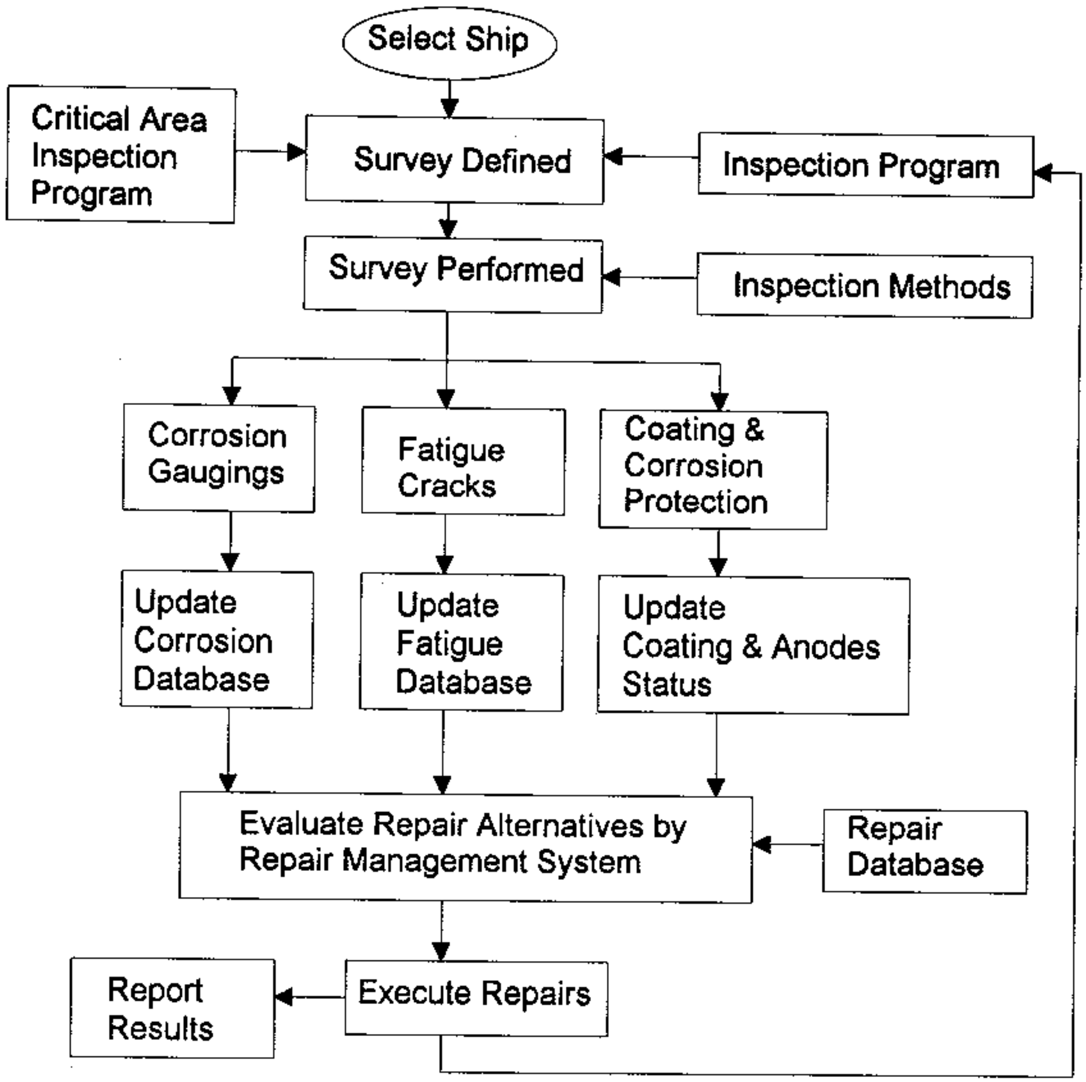


Figure 6.6: Basic parts of RMS system for inspection, maintenance, & repair

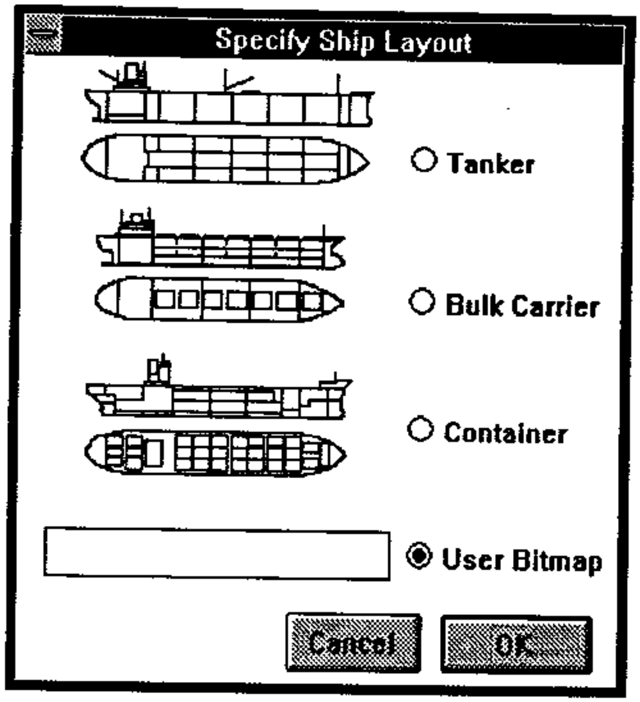


Figure 6.7: Three pre-defined typical ship layouts can be chosen.

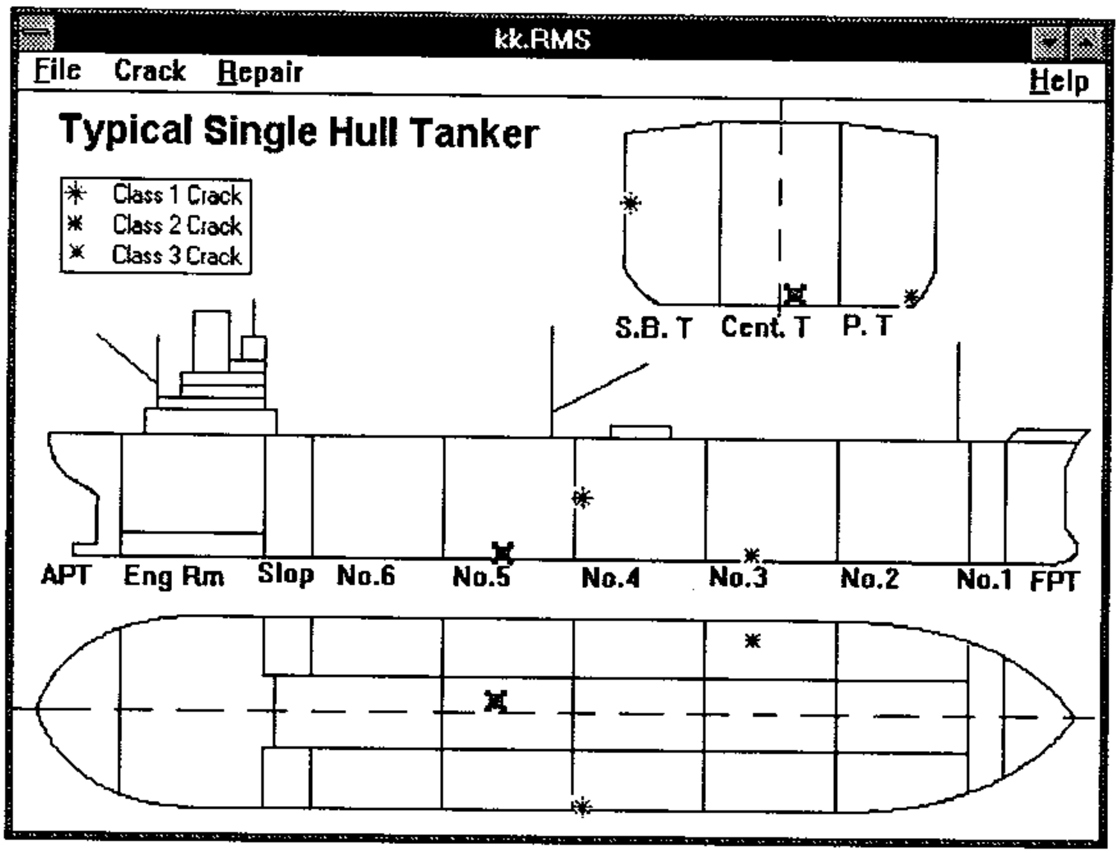


Figure 6.8: Three cracks are inputted into a tanker layout.

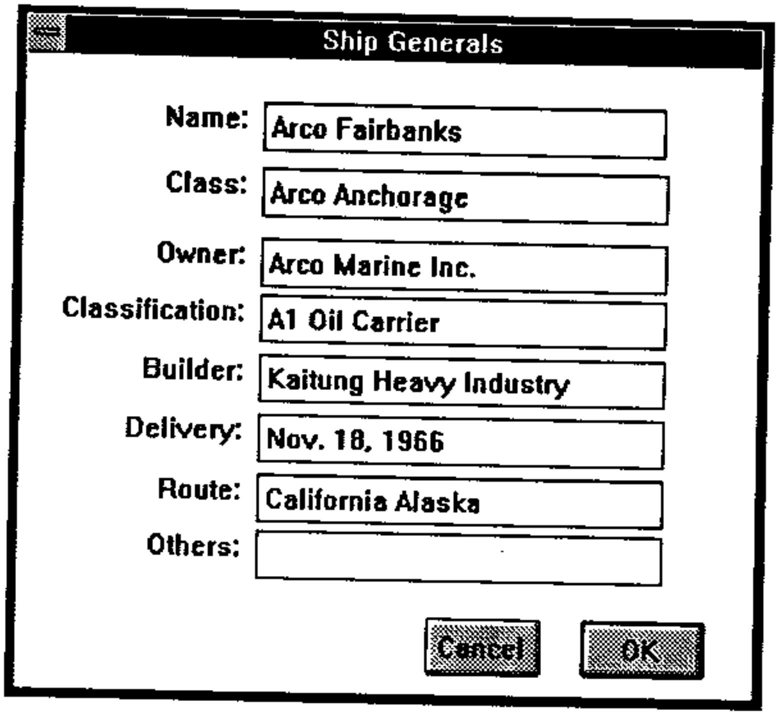


Figure 6.9: Input ship general information.

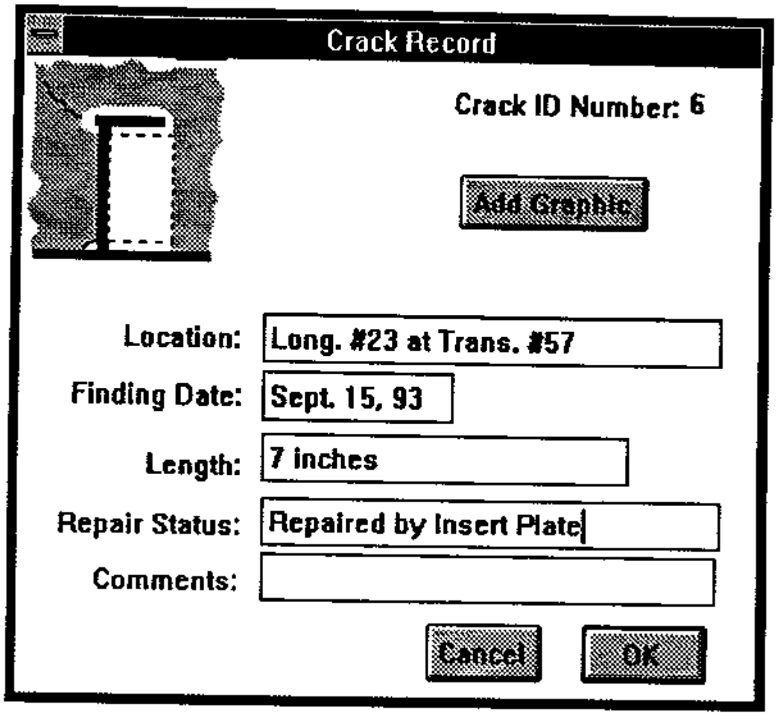


Figure 6.10: This shows a crack record with an attached graphic.

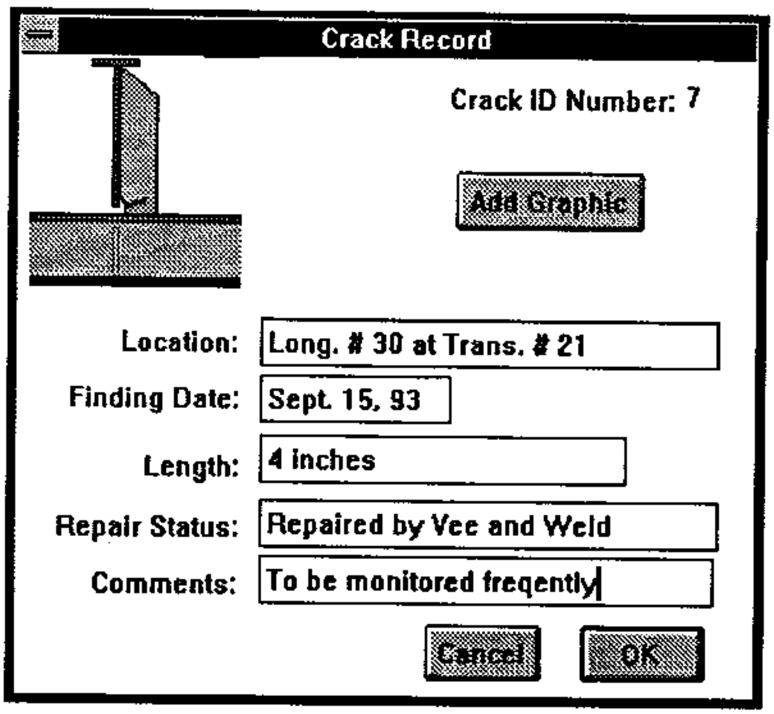


Figure 6.11: This shows an another crack record with an attached graphic.

radius (corner) should have an S-N curve very close to F2-class (the un-cracked cutout radius has a C-class curve). This is not a good result because a sound butt weld on a flat plate is usually assigned a D-class curve. The possible reason for this poor result is the lack of sufficient data.

On the other hand, the second case with a much greater database produces a very reasonable result. It suggests that re-welding in flatbar welds should have an S-N curve slight lower than an F2-class curve (the original detail has an F2-class).

This method assumes that very high quality inspections are carried out that all significant cracks are found. The quality of the results is highly depended on the inspection data collected. Most of the current survey reports simply record cracks in a very rough format, they usually do not specify the detail crack location. Most existing survey reports do not provide sufficient information. Also, it is difficult to locate a sufficient crack database. However, given sufficient high quality inspection information on fatigue cracking, this method does provide an alternative to define S-N curves.

This project also carried out 24 cases of FEA on local CSD models. The percentage of stress reduction due to a certain repair by geometry change is defined as Stress Reduction Factors (SRF). If these factors were known, a repair analysis could be done in very little time by using the RMS program. In order to obtain these SRFs, FEA models of CSDs before repairs are compared with models of CSDs after repairs. A table of the SRFs for typical repairs was developed.

There were problems in defining load ratios when applying load conditions to the models. The load can be generally broken into four components (vertical bending, horizontal bending, pressure and shear). The load ratio is dependent on the location in a ship. In order to eliminate this dependency, only unit pressure was applied to these models. This means that these SRFs can be used only in those failures that are mainly due to cyclic pressure loadings. For more comprehensive results, a complete FEA loading analysis of the CSD is needed.

The case study performed on the repair of a transverse cutout failure on the side shell structure using the RMS program clearly demonstrates the usefulness of the RMS. The RMS can quickly perform a comparative analysis of repairs. With proper information on the available repair alternatives, CSDs and costs, consistent repair decisions can be made quickly.

7.2 Conclusions

The results from this project illustrates that despite the complexities of the repair analysis process, the RMS can be used to significantly improve and simplify the traditional approach. With the new approach developed in the RMS, it is possible to make quick, intelligent repair decisions for the repair of ship CSDs. The RMS outlined in this project can be developed into a powerful tool to aid repair engineers in fatigue crack repair analysis. This development effort should include:

- development and maintenance of a complete library of details that represent both old and current designs;
- structuring the finite element analysis results in the RMS stress reduction factor format for quick repair analysis;
- tuning of the load ratios or the development of a new system to determine relative loads;
- development of a sophisticated database system to easily manage the input data;
- continued verification of the RMS system.

To implement the complete RMS concept, significant effort and a long-term commitment are required. This effort would involve all phases of repair analysis and require professional programmers to work with naval architects. High priority in this effort should be placed on proper knowledge representation in ship structural maintenance and repair.

7.3 Future Directions and Recommendations

The repair of ships has been used as a basis to illustrate the possible application of computer technology to help solve a difficult engineering problem. The scope of the current work was constrained and limited due to the time available. As a result, many enhancements to the RMS program and the current research are possible. The high priority future directions are outlined in the following parts of this chapter.

7.3.1 Investigate Load Ratios

In Chapter 5, a series of FEA models were analyzed in order to calculate stress reduction factors (SRF). A load condition musts to be defined and applied to the models. The load can be broken down into four components (vertical bending, horizontal bending, pressure and shear). Since a linear FEA is used, the four loads can be applied separately to the models. Then, the resulting stress (or stress reduction factor) can be superimposed. However, the ratio of these four loads is unknown and it is dependent on the locations in ships. This may be determined by performing a global FEA. Once the ratio is found as a function of location, stress reduction factors due to each load can be combined. A more realistic SRF can be determined.

... based on FEA using pressure loadings. It

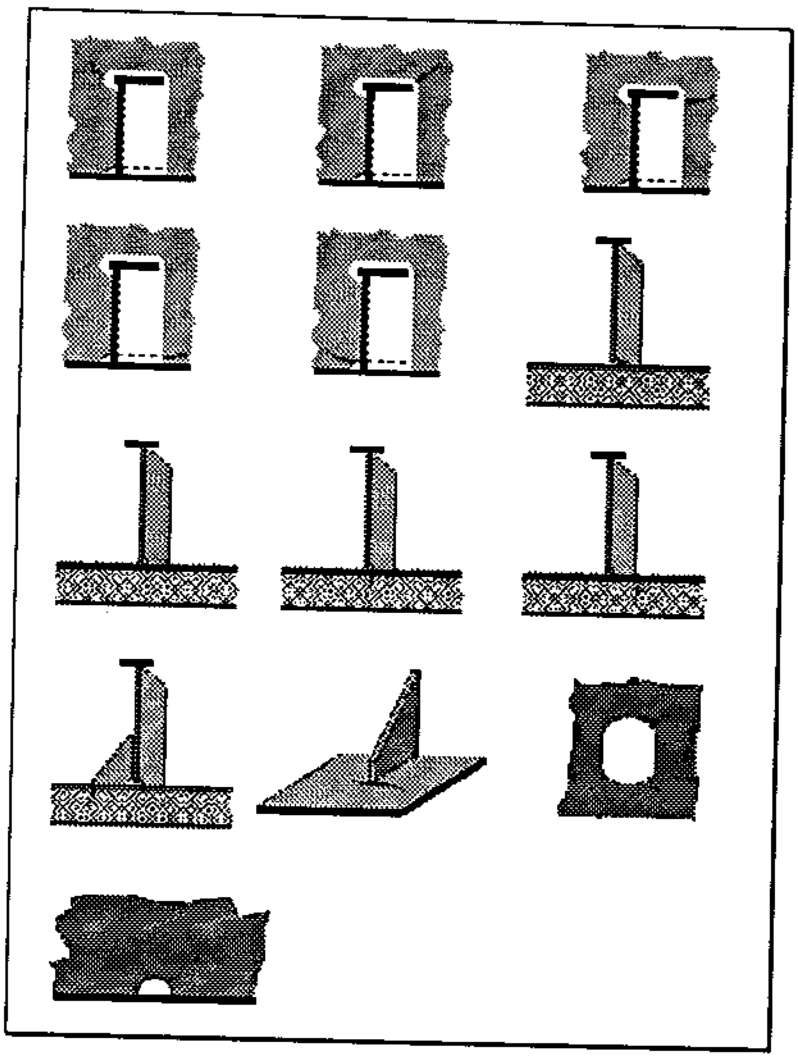


Figure 6.12: There are 13 types of pre-defined cracked structural details.

assessment. Among these, the ship condition assessment is probably the most important, and more appropriate to the RMS style of analysis. Ship condition assessment is directly related to the ship condition database and could prove invaluable to owner/operators, classification societies, and ship repair yards in their efforts to monitor fleets of aging ships.

7.3.4 Improve RMS Program Features

The current RMS has a crack management database in which users can record cracks on graphical ship layouts. Statistical functions can be added to this database to allow users evaluate trends or distributions of cracks. Since inspection is such a monumental task on crude oil carriers and other large ships, the RMS could be expanded to help inspectors to ship locations with the highest probability of failure. This ability would be closely tied to a reliability analysis of the entire ship structure and a tracking of the failure probabilities for all components. Continuous updating of the failure probabilities using historical data or instrumentation is possible. Updated failure probabilities could be used directly for repair analyses.

Another area that can be improved is the method of graphical ship layouts. The crack database is organized through a fixed three-view ship layout that cannot be zoomed in or out. When a large number of cracks are input, the layout becomes a 'dense cloud' of fatigue cracks. It is possible to rebuild the program such that the fixed layout can be zoomed in and the selected tank drawing will popped up to give users a more detailed picture.

A handy feature that can greatly improve the use of the RMS is to add the Print function. With a Print function, inspectors can print out a ship layout with all the cracks found previously before a hull inspection. This will give inspectors a clear idea on where the cracks may locate. Also repair engineers can more easily evaluate the condition of a ship through the printout.

Another feature that can be improved is expanding the Help function to provide a clear explanation facility to teach the users of the RMS about repair analysis. This could be a valuable for training tool for repair personnel. There are two Help files in the current RMS, one is to teach users how to use the RMS program. The other one is to provide general information of ship structural maintenance and repair including graphical repair examples, steel repair, maintenance of corrosion protection system and others. More Help files can be easily added within the Help command in the Windows menu bar.

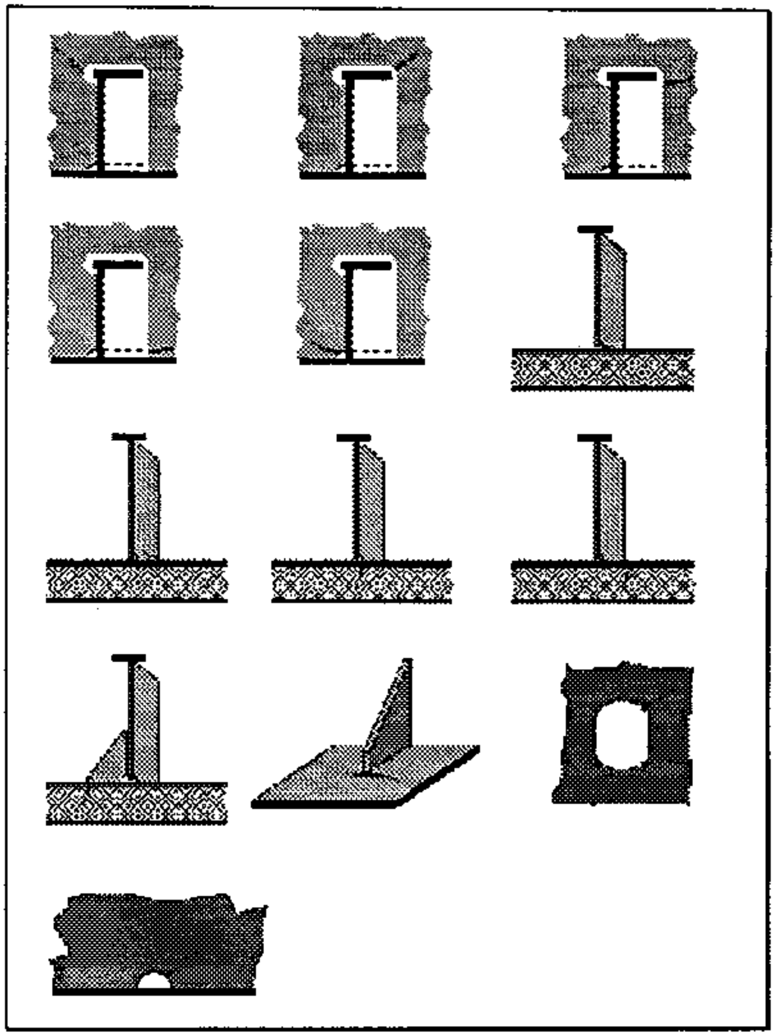


Figure 6.12: There are 13 types of pre-defined cracked structural details.

301

The SRFs established in this study are based on FEA using pressure loadings. It means these SRFs are only suitable for fatigue cracks due to cyclic pressure like those in side shell structure and in the forepeak. While SRFs are established from four loads with a correct ratio, the RMS approach could be expanded to ship components other than those whose cyclic loadings are dominated by water/cargo pressures.

7.3.2 Continue S-N Calibration Based on Inspection Data

During this project, S-N calibrations for two major types of repairs were carried out in two case studies. The results indicate that repaired details have very similar S-N classification (but slight lower) to normal details with similar geometry. It would be worthwhile to continue collecting inspection data. With new data, the calibrated S-N curves can be continually updated. Although it is very hard to collect enough data on crack and re-cracking, this data collecting can be done through a special long-term program developed for this purpose.

7.3.3 Add More Failure Modes

Failure mode and cause analysis are an obvious area for future improvement. A majority of ship cracking failures, especially in crude oil carriers, are clearly due to fatigue. As a result, detailed mode and cause analysis are not currently as important as evaluating fatigue failures. However as ship designs change, new modes and causes of failure may develop. A tool to helping evaluate these new modes and causes could prove to be important.

Fatigue cracking is not the only mode of failure in ships, but the most common. Other important modes include buckling, corrosion, global strength, and ship condition

CHAPTER 7. SUMMARY AND FUTURE DIRECTIONS

7.1 Summary

A framework for the development of a Repair Management System (RMS) to aid in ship structural repair evaluation has been developed. The RMS is the first known attempt to handle the complexities of ship structural repair analysis in a framework that provides both quick decisions and thorough evaluations.

The most important development in this study is an improved approach to fatigue life estimation for repaired structural details. As is well known, one of the most difficult parts involved in a fatigue analysis is load computation. Two parameters (Weibull parameter and extreme stress) need to be determined before computing the fatigue damage. This project has developed a new approach to determine these two parameters. The approach uses the value of the original fatigue life to solve for the extreme stress while assuming an arbitrary Weibull parameter. It produces amazingly good consistency on the estimated repair lives. This new approach makes an on-site repair analysis possible in repair yards. The limitation of this approach is that it can be used on only repaired details because the original fatigue life is a crucial input.

Beside load computation, the S-N classification for CSDs is another existing problem with fatigue analysis. The S-N designation is usually done by engineering judgment [Chen, 1992]. This research also explored a new method to calibrate S-N data for repaired details. This method considers the structural details in an in-service ship as full scale specimens. While a large number of re-cracking data on a particular detail are collected, the S-N curve corresponding to the repaired detail can be found. Two case studies were completed. The first case suggests that the veeing and welding on a cutout

REFERENCES

Almar-Naess, A. (ed.), Fatigue Handbook, Offshore Steel Structures, Chapter 6, "Improving the Fatigue Strength of Welded Joints", Tapir, 1985.

Bea, Robert G., "Marine Structural Integrity Programs", Ship Structure Committee Report No. SSC-365, 1992.

Chen, Y.K., "Fatigue Classification of Ship Structural Details", SMP Report No. 1-4, American Bureau of Shipping, August 1992.

Chevron, 'Chevron California - Survey Report", 1972, 1981, 1982, 1984, 1986, 1987, 1990, 1991, 1992 and 1993.

Committee III.1 Report, Proceedings of the Seventh International Ship Structures Congress, Paris, 1979

DnV, "Inspection, Monitoring, Maintenance/Repair", Report of Committee V.2, 11th International Ship & Offshore Structures Congress, China, 1991.

Fisher, J.W., Dexter, R.J. and others, 'Structural Failure Modes for Advanced Double Hull - Fatigue and Fracture Failure Modes", Fleet of the Future, Lehigh University, 1993.

Gallion, Keith A. and Bea, Robert G., "Repair Management System, A System to Aid in the Diagnosis of Ship Structural Failures and the Evaluation of Repair Alternatives", Structural Maintenance Project for New and Existing Ships, Report No. SMP 4-1, Dept. of Naval Architecture & Offshore Engineering, U.C. Berkeley, June 1992.

ISSC, "Fatigue and Fracture", Proceedings of the Tenth International Ship and Offshore Structures Congress, Volume 1, Lyngby, August, 1988.

ISSC, "Fatigue and Fracture", Proceedings of the Eleventh International Ship and Offshore Structures Congress, 1991.

Jordon, C. R and Cochran, C. S., Newport News Shipbuilding, "In Service Performance of Structural Details", Ship Structural Committee Report No. SSC-272, 1978

Jordon, C. R. and L. T. Knight, "Further Survey of in-Service Performance of Structural Details", Ship Structural Committee Report SSC-294, Newport News Shipbuilding, 1980.

Ma, Kai-tung and Bea, Robert G., "Durability Considerations for New & Existing Ships", Structural Maintenance Project for New and Existing Ships, Report No. SMP 5-1, Dept. of Naval Architecture & Offshore Engineering, U.C. Berkeley, September 1992.

Ma, Kai-tung and Bea, Robert G., "RMS--Repair Management System (Further Development)", SMP Report No. 4-2, Department of Naval Architecture & Offshore Engineering, University of California, Berkeley, September 1993.

Morrill, J. P. and D. Wright, "A Method for Reasoning By Analogy in Failure Analysis", Transactions of the American Society of Mechanical Engineers, Volume 111, July, 1989.

Morrill, J. P. and D. Wright, "A Model of Categorization for Use in Automated Failure Analysis", Journal of Vibration, Acoustics, Stress, and Reliability in Design, Volume 110/559, October, 1988.

Munse, W.H., Thomas W. Wilbur, Martin L. Tellalian, Kim Nicoll and Kevin Wilson, "Fatigue Characterization of Fabricated Ship Details for Design", Ship Structural Committee Report SSC-318, Department of Civil Engineering, University of Illinois at Urbana-Champaign, 1983.

Ochi, Michel K., "Applied Probability and Stochastic Processes - In Engineering and Physical Sciences", John Wiley & Sons, 1990.

Roddis, W. M. Kim and Jerome Connor, "Qualitative/Quantitative Reasoning for Fatigue and Fracture in Bridges", Coupling Symbolic and Numerical Computing in Expert

Systems, II, J.S. Kowalik and C.T. Kitzmiller (ed.), Elsevier Science Publishers B. V. (North-Holland), 1988.

Schulte-Strathaus Rolf and Bea, Robert G., "Fatigue Database Development and Analysis", Project Report No. SMP 4, Structural Maintenance for New and Existing Ships, Dept. of Naval Architecture & Offshore Engineering, U.C. Berkeley, 1991.

Schulte-Strathaus Rolf and Bea, Robert G., "Development of Calibrated S-N Curves and System for the Selection of S-N Curves", Project Report No. FACTS 1-1, Structural Maintenance for New and Existing Ships II, Dept. of Naval Architecture & Offshore Engineering, U.C. Berkeley, 1993.

Schulte-Strathaus Rolf and Bea, Robert G., "Ship Structural Integrity Information System (SSIIS) - Development of an Integrated Vessel Database System", Project Report DTMA-91-93-G-00041, Dept. of Naval Architecture & Offshore Engineering, U.C. Berkeley, 1994.

Stambaugh, KA, Wood, WA, "A Non-Expert's Guide for inspecting and Determining the Causes of Significant Ship Fractures" Ship Structure Committee, SSC-337, 1987.

Stambaugh, Karl A. and William A. Wood, "Ship Fracture Mechanisms Investigation", Ship Structural Committee Report SSC-337 (Part 1 and 2), Giannotti and Associates, Inc., 1990

Stear, James and Paulling, J. R., "Structural Analysis and Loadings", Project Report No. SMP 3-1, Structural Maintenance for New and Existing Ships, Dept. of Naval Architecture & Offshore Engineering, U.C. Berkeley, 1992.

Tikka, Kirsi K., Chevron Shipping, "Inspection and Structural Maintenance of Chevron Double Hull Tankers", 1991.

TSCF, (Tanker Structure Co-operative Forum), "Guidance Manual for the Inspection and Condition Assessment of Tanker Structures", 1986.

TSCT, (Tanker Structure Co-operative Forum), "Condition Evaluation and Maintenance of tanker Structures", 1992.

USCG, "Report on the Trans-Alaska Pipeline Service (TAPS) Tanker Structural Failure Study", Office of Marine Safety Security and Environmental Protection, Washington, D.C, June 25, 1990.

Wirsching, Paul H., 'Probability Based Fatigue Design Criteria for Offshore Structures", Final Report, American Petroleum Institute, PRAC Project 81-15, 1983.

Wirsching, Paul H. and Y.-N. Chen, "Considerations of Probability-Based Fatigue Design for Marine Structures", Society of Naval Architects and Marine Engineers, paper presented to Marine Structural Reliability Symposium, Arlington, Virginia, October 5-6, 1987.

APPENDIX A. SOURCE CODES

A.1 Codes

Due to the large amount of source code, only one module of the source code is listed here. The remaining modules are documented in a separate report [Ma & Bea, 1993]. This module is the fatigue analysis module.

The necessary information to conduct the repair analysis is provided by interactive input from the user and pre-defined data in the program. Ship loading information, including the Weibull parameter, average stress frequency, and expert load zones and ratios are pre-defined in the program. Stress reduction factors for each configuration, and S-N class designations for each location are pre-defined, too. Interactive inputs includes detail configuration and failure location, the mean time to failure of the original detail and the desired repair option. With all the information above, the program calculate the expected repair life by using the approach developed in Chapter 3. Repair analysis is conducted only at the location of failure.

```
/*******************************************************

Module:    CalcuFat.c

Revision: July 9, 1994 by Kai-tung Ma

PURPOSE:  Calculate fatigue life.

FUNCTIONS:
          CalcFatLife(float   T1, Weib,
                              A,    m,    dm,     Sq,
                              newA, newm, newdm, newSq,
                              r)

LOCAL:    gammln()
          ingamm()
          gser()
```

```
        gcf()
        TwoS_S-N()
        StressS0()
        ModifyS0()
        FatigueLife()

**********************************************************/
#include <math.h>
#include "Repair.h"

#define ITMAX  100
#define EPS    3.0e-7
#define f0     2500000         /* Cycle per Year */
#define Df     1               /* Damage Factor */
#define B      1               /* Uncertainty */

/**********************************************************

FUNCTION: float gammln(float)

PURPOSE:  Return the value ln[gamma(xx)] for xx>0.  Full
accuracy is btained for xx>1.

Source:   This function is adapted from "Numerical Recipes
in C - The art of Scientific Computing", William H. Press,
Brian P. Flannery, Saul A. Teukolsky, William T.
Vetterling.

**********************************************************/
float gammln(xx)
float xx;
{
     double x, tmp, ser;
     static double cof[6]={76.18009173, -86.50532033,
                   24.01409822, -1.231739516,
                   0.120858003e-2, -0.536382e-5};
     int j;

     x=xx-1.0;
     tmp=x+5.5;
     tmp -= (x+0.5)*log(tmp);
     ser=1.0;
     for(j=0;j<=5;j++) {
          x += 1.0;
          ser += cof[j]/x;
          }
     return -tmp+log(2.50662827465*ser);
}

/**********************************************************
```

```
FUNCTION: float ingamm(float, float)

PURPOSE:  Returns the incomplete gamma function r(a, x).

Source:   This function is revised from "Numerical Recipes
in C: The Art of Scientific Computing", William H. Press,
Brian P. Flannery, Saul A. Teukolsky, William T.
Vetterling.

********************************************************/
float ingamm(a, x)
float a, x;
{
     float gamser, gammcf, gln;
     void gser(), gcf();

     if(x<0.0 || a<= 0.0) /*nrerror("Invalid arguments in
                                    routine GAMMP()")*/;
     if(x< (a+1.0)) {
          gser(&gamser, a, x, &gln);
          return (gamser * exp(gln));
     } else {
          gcf(&gammcf, a, x, &gln);
          return ((1.0-gammcf) * exp(gln));
     }
}

/********************************************************

FUNCTION: float gser(float, float, float, float)

PURPOSE:  Returns the incomplete gamma function P(a, x)
evaluated by its series representation as gamser.  Also
returns gammln() as gln.

Source:   This function is adapted from "Numerical Recipes
in C: The Art of Scientific Computing", William H. Press,
Brian P. Flannery, Saul A. Teukolsky, William T.
Vetterling.

********************************************************/
void gser(gamser, a, x, gln)
float a, x, *gamser, *gln;
{
     int n;
     float sum, del, ap;
     float gammln();
     // void nrerror();
```

```
      *gln=gammln(a);
      if(x <= 0.0) {
           /*if(x < 0.0) nrerror("x less than 0 in routine
                                  GSER()"); */
           return;
      } else {
           ap=a;
           del=sum=1.0/a;
           for (n=1;n<=ITMAX;n++) {
                ap += 1.0;
                del *= x/ap;
                sum += del;
                if (fabs(del) < fabs(sum)*EPS) {
                     *gamser=sum*exp(-x+a*log(x)-(*gln));
                     return;
                }
           }
           /* nrerror("a too large, ITMAX too small in
                                  routine GSER()"); */
           return;
      }
}

/**********************************************************

FUNCTION: float gcf(float, float, float, float)

PURPOSE:  Returns the incomplete gamma function Q(a, x)
evaluated by its continued representation as gammcf.  Also
returns gammln() as gln.

Source:   This function is adapted from "Numerical Recipes
in C: The Art of Scientific Computing", William H. Press,
Brian P. Flannery, Saul A. Teukolsky, William T.
Vetterling.

**********************************************************/
void gcf(gammcf, a, x, gln)
float a, x, *gammcf, *gln;
{
      int n;
      float gold=0.0, g, fac=1.0, b1=1.0;
      float b0=0.0, anf, ana, an, a1, a0=1.0;
      float gammln();
//    void nrerror();

      *gln=gammln(a);
      a1=x;
      for (n=1;n<=ITMAX;n++) {
           an=(float) n;
```

```
            ana=an-a;
            a0=(a1+a0*ana)*fac;
            b0=(b1+b0*ana)*fac;
            anf=an*fac;
            a1=x*a1+anf*a1;
            b1=x*b0+anf*b1;
            if (a1) {
                 fac=1.0/a1;
                 g=b1*fac;
                 if (fabs((g-gold)/g) < EPS) {
                      *gammcf=exp(-x+a*log(x)-(*gln))*g;
                      return;
                 }
                 gold=g;
            }
      }
      /* nrerror("a too large, ITMAX too small in routine
                                                   GCF()"); */
}

/*******************************************************

FUNCTION: float TwoS_SN(float, float)

PURPOSE:  Calculate a modification factor which accounts
two slope SN curve.

*******************************************************/
float TwoS_SN(float m, float dm, float Sq, float S0, float
FatLife, float e)
{
      float u, v;

      v = pow( Sq/S0, e) * log( f0*FatLife);

      u = 1 - 1/exp( gammln(1+m/e))
           * ( ingamm(1+m/e, v) - pow(v, -dm/e) *
           ingamm(1+(m+dm)/e, v) );
      return u;
}

/*******************************************************

FUNCTION: float StressS0(float, float, float, float)

PURPOSE:  Calculate extreme stress 'S0'

*******************************************************/
float StressS0(float Tf, float Wb, float A, float m, float
Dm, float Sq)
```

```
{
      float S0, tS;

      S0 = pow(log(f0*Tf), 1/Wb)
            / B
            * pow(Df*A/f0/Tf
            / exp(gammln(m/Wb+1)), 1/m);


      do    {
            tS = pow(log(f0*Tf), 1/Wb)
                  / B
                  * pow(Df*A/f0/Tf
                  / TwoS_SN(m, Dm, Sq, S0, Tf, Wb)
                  / exp(gammln(m/Wb+1)), 1/m);

            S0 = pow(log(f0*Tf), 1/Wb)
                  / B
                  * pow(Df*A/f0/Tf
                  / TwoS_SN(m, Dm, Sq, tS, Tf,Wb)
                  / exp(gammln(m/Wb+1)), 1/m);

      } while((S0-tS)>0.01 || (S0-tS)<-0.01 );


      return S0;
}

/*********************************************************

FUNCTION: float ModifyS0(float, float)

PURPOSE:
Calculate a modification factor to revise the extreme
stress S0 into S0'.

*********************************************************/
float ModifyS0(float Tf1, float Tf2, float Wb)
{
      float factor;

      factor = pow( log(f0*Tf2/0.632120558)
                / log(f0*Tf1/0.632120558) , 1/Wb );
      return factor;
}

/*********************************************************

FUNCTION: float FatigueLife(float, float, float, float,
float)
```

```
PURPOSE:  Calculate expected fatigue life

*****************************************************/
float FatigueLife(float S0, float Tf, float Wb,
                float NewA, float Newm, float NewDm,
                float NewSq)
{
     float Life;

     Life = Df
          * NewA
          * pow(log(f0*Tf), Newm/Wb)
          / f0
          / pow( (B*S0), Newm)
          / TwoS_SN(Newm, NewDm, NewSq, S0, Tf, Wb)
          / exp(gammln(Newm/Wb+1));

     return Life;
}

/*****************************************************

FUNCTION: float CalcFatLife()

PURPOSE:
Process S-N data and detail types etc. in order to compute
fatigue life.

MESSAGES:

*****************************************************/
float CalcFatLife(float T1, float Weib, float A, float m,
float dm, float Sq, float newA, float newm, float newdm,
float newSq, float r)
{
     float S1, S2, T2, tempL2, tL;

     S1 = StressS0(T1, Weib, A, m, dm, Sq);

     T2 = FatigueLife(S1, T1, Weib, newA,
                         newm, newdm, newSq);

     do {
          tL = T2;
          S2 = S1 * ModifyS0(T1, T2, Weib);

          S2 = S2 * r;
```

```
            T2 = FatigueLife(S2, T1, Weib, newA,
                                    newm, newdm, newSq);
            do {
                tempL2 = FatigueLife(S2, T2, Weib,
                                newA, newm, newdm, newSq);
                T2 = FatigueLife(S2, tempL2, Weib,
                                newA, newm, newdm, newSq);
            } while((T2-tempL2)>0.01 || (T2-tempL2)<-
                                                    0.01);

        } while((T2-tL)>0.01 || (T2-tL)<-0.01);

        return T2;
    }
```

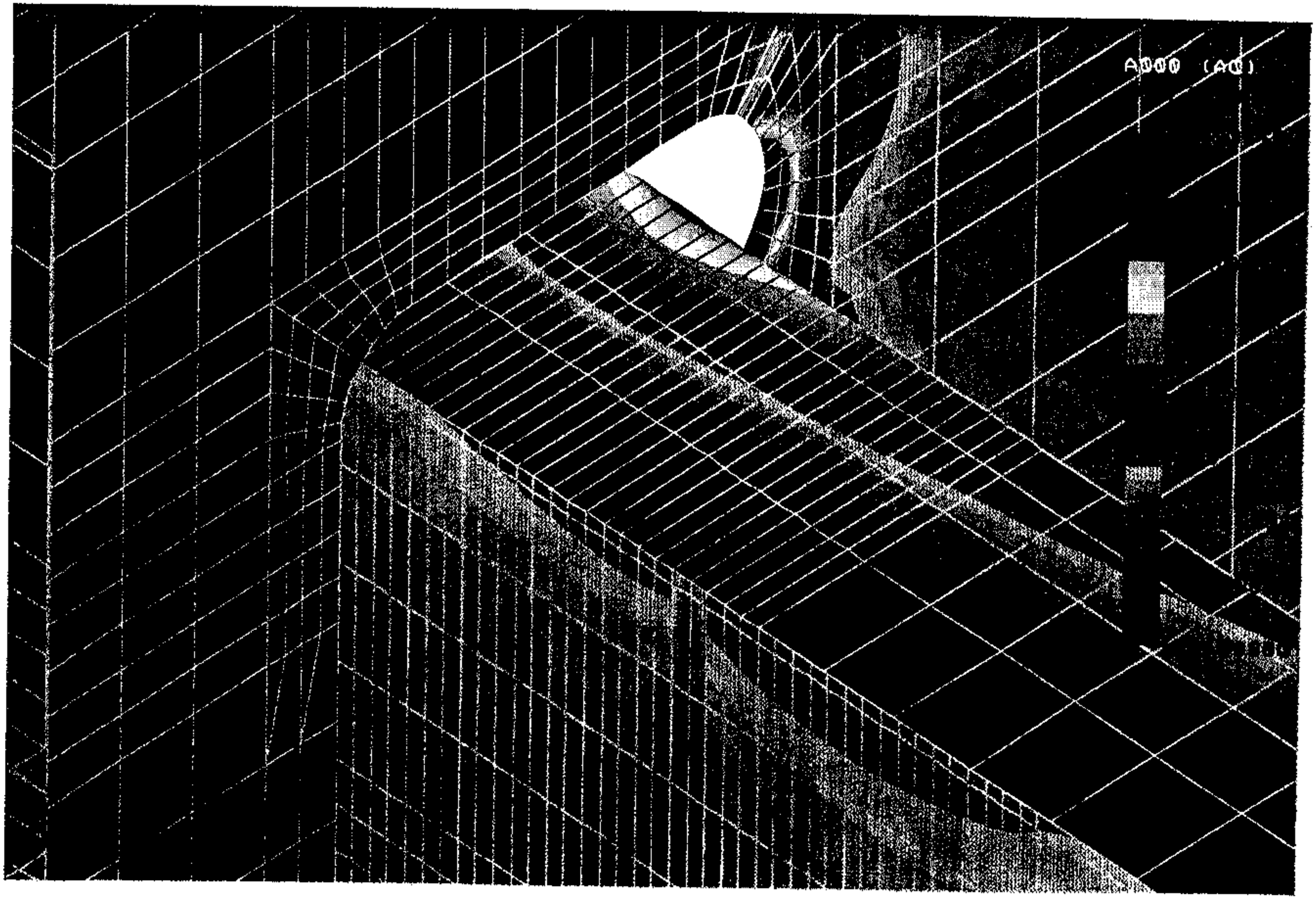
APPENDIX B. FINITE ELEMENT MODELS

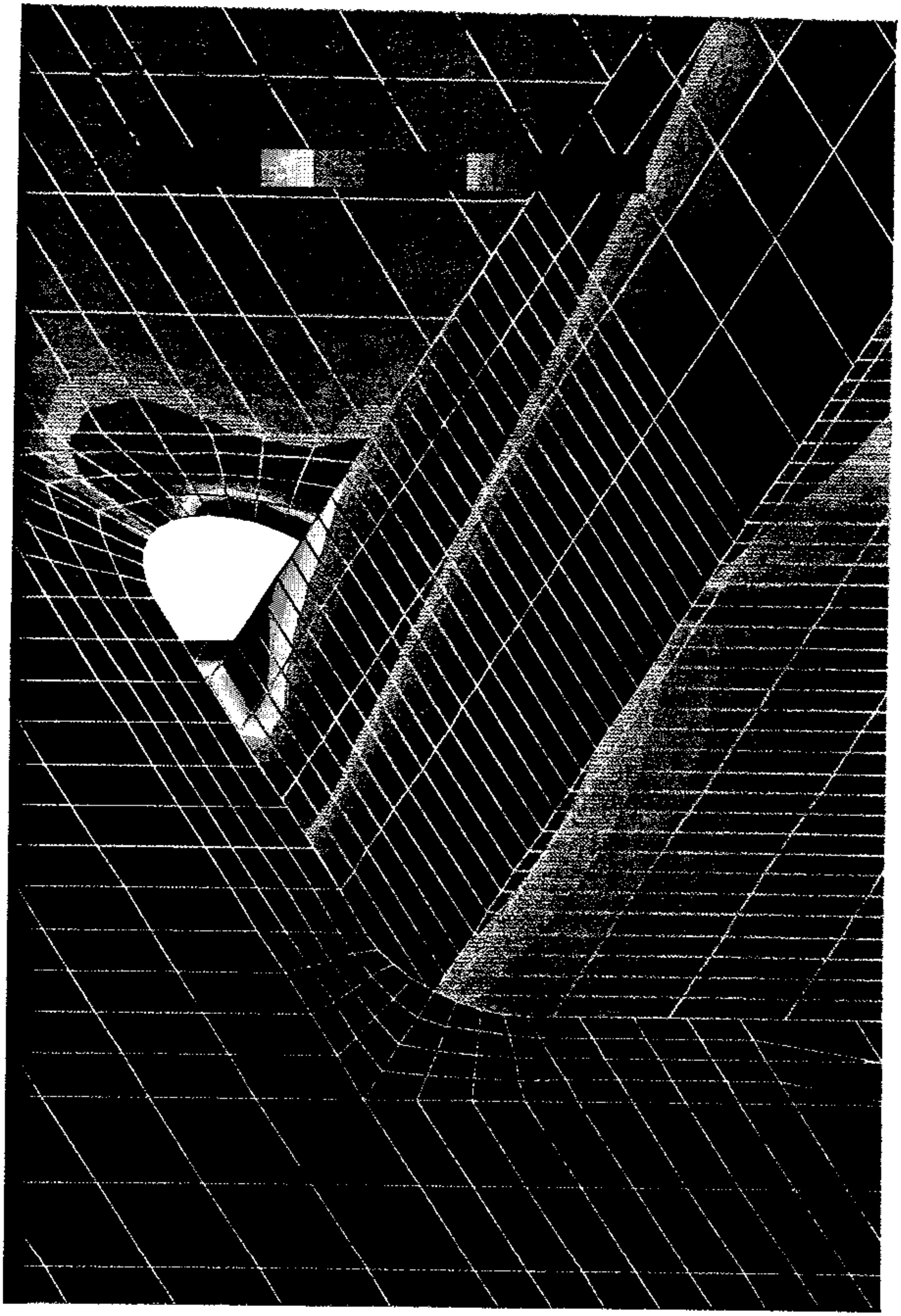
Included in this Appendix are the finite element models and results. The models were created and analyzed by using the linear finite element program. All models were built by using shell elements. The models represent local structures in side shell of tankers. They are rotated 90 degree so side shells are on the FEA model bottoms. Their scantlings are based on three tankers. See Chapter 5 for more details on these CSDs.

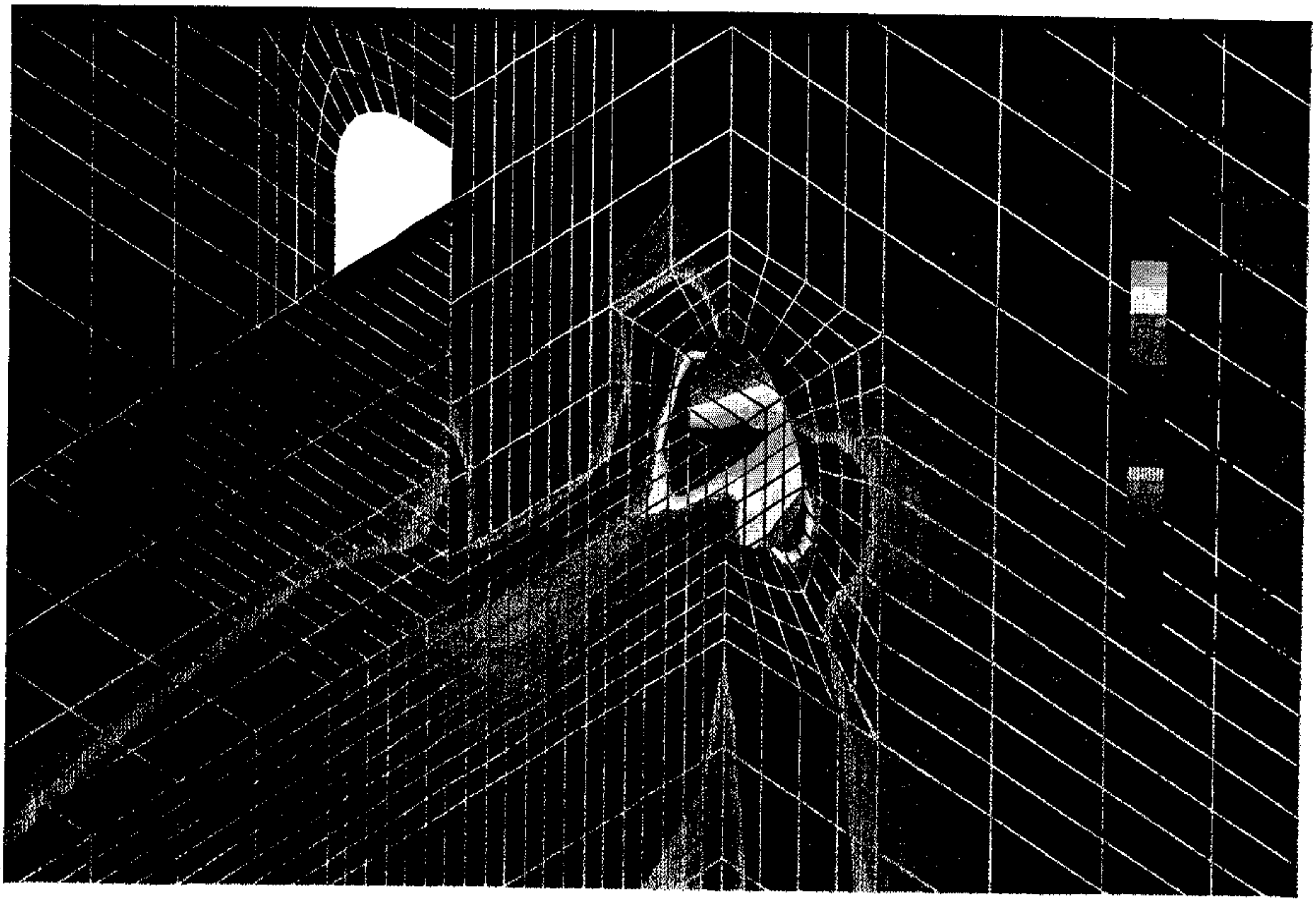
B.1 CSD #1 (model Axxx)

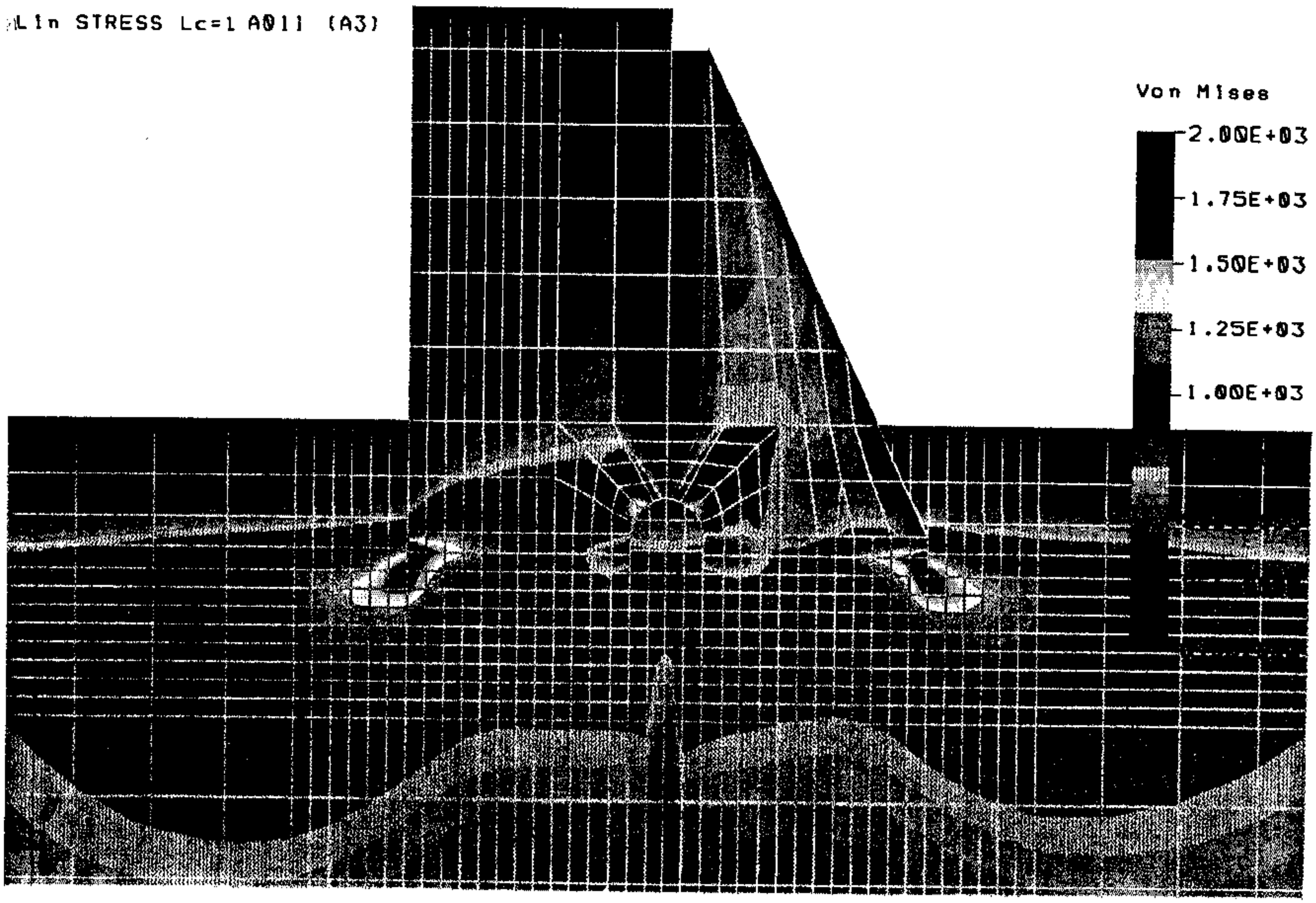
The following pages show the model meshing and the stress contours for the 8 models based on CSD #1. The 8 models are:

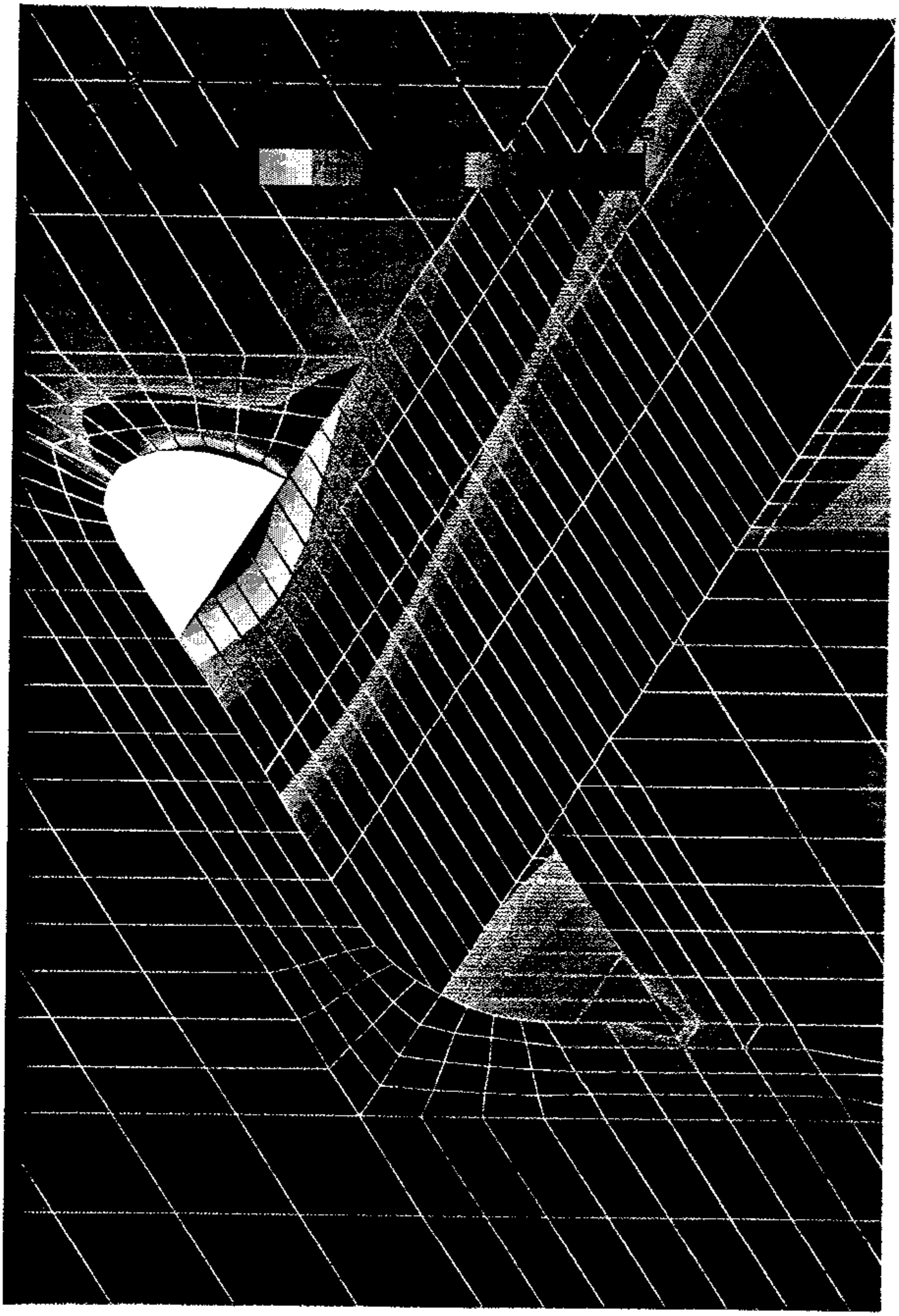
- A000 (A0) is a basic CSD.
- A001 (A1) is a basic CSD with bracket.
- A010 (A2) is a basic CSD with flatbar.
- A011 (A3) is a basic CSD with flatbar and bracket.
- A100 (A4) is a basic CSD with lug.
- A101 (A5) is a basic CSD with lug and bracket.
- A110 (A6) is a basic CSD with lug and flatbar.
- A111 (A7) is a basic CSD with lug, flatbar and bracket.

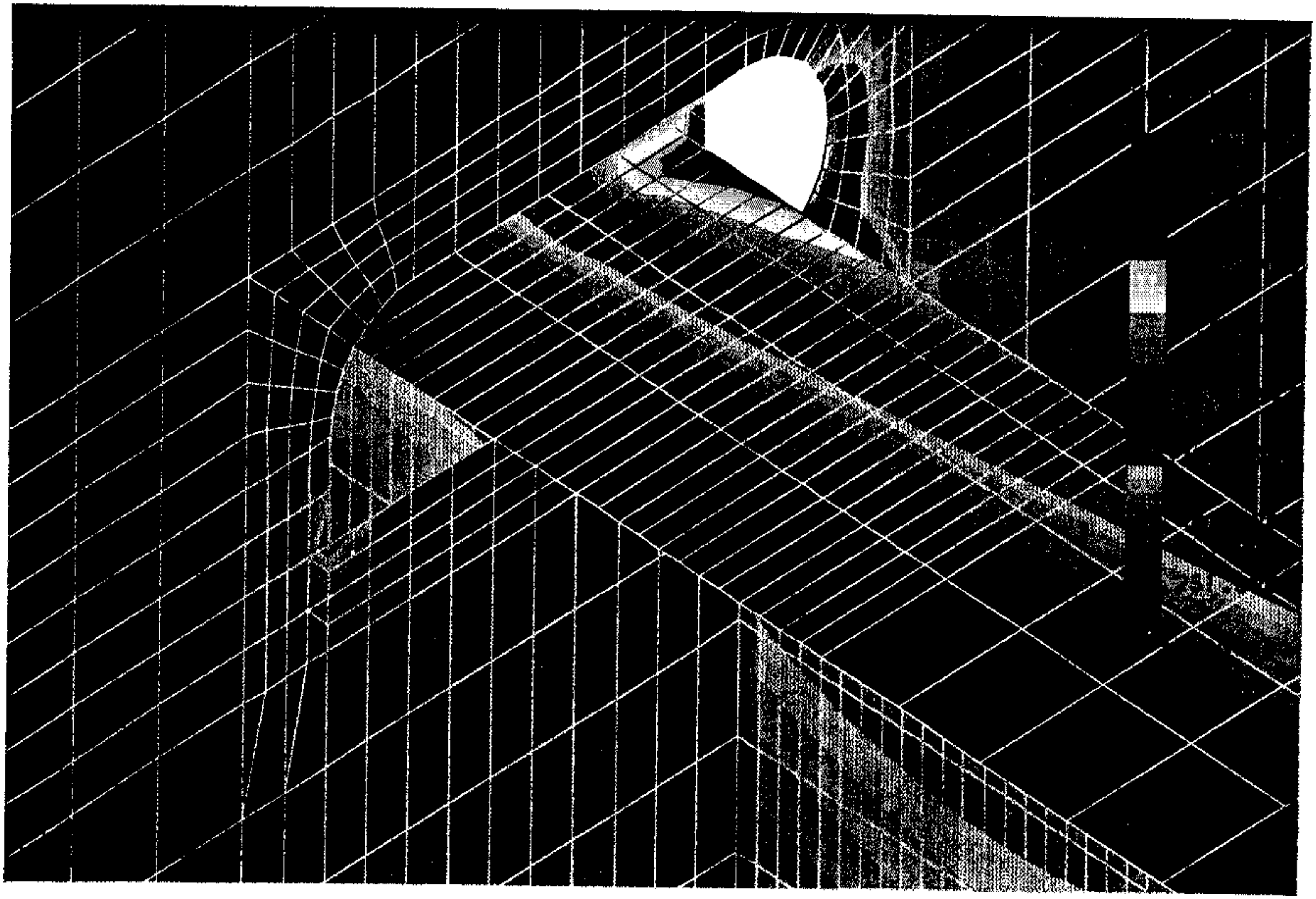

A000

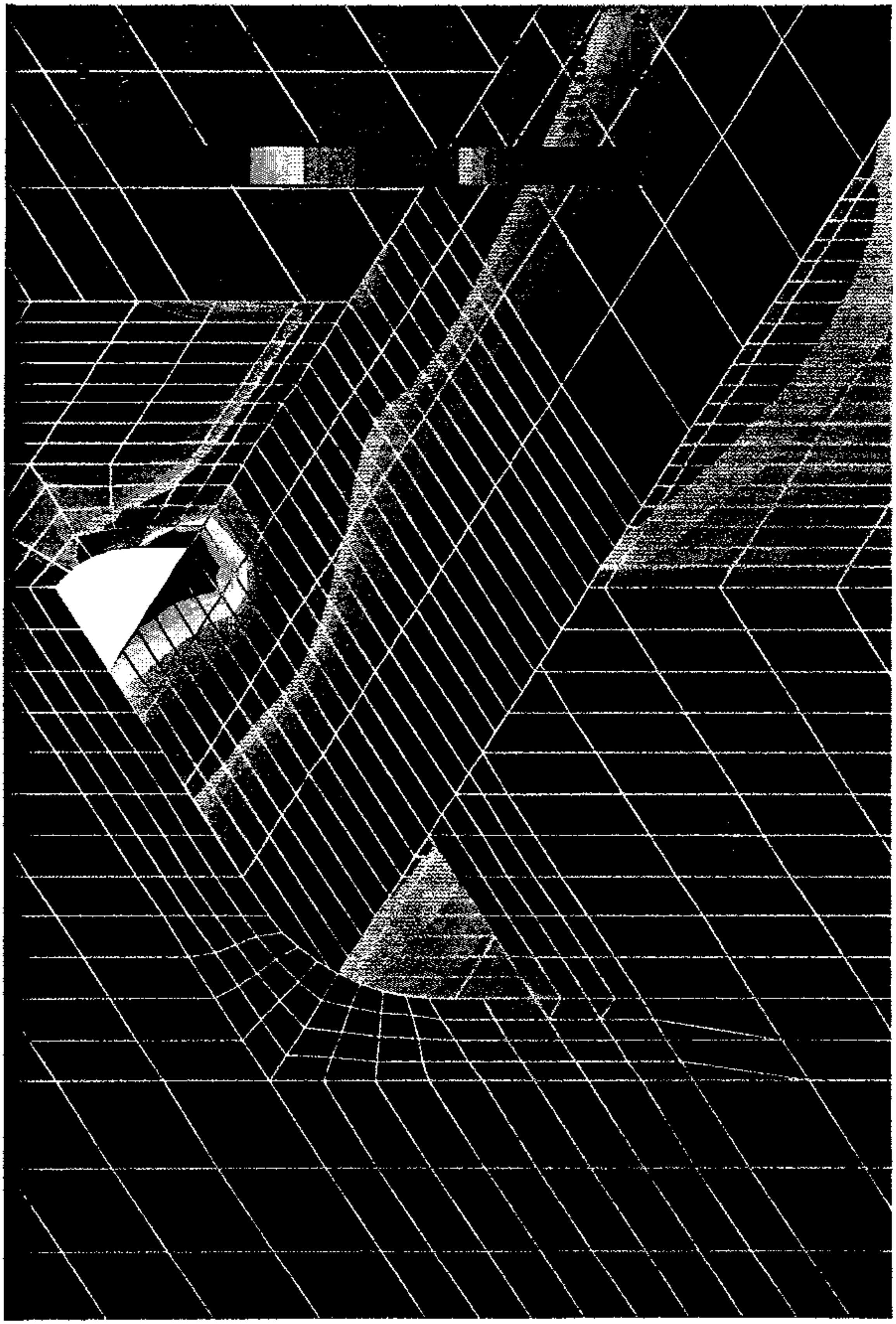


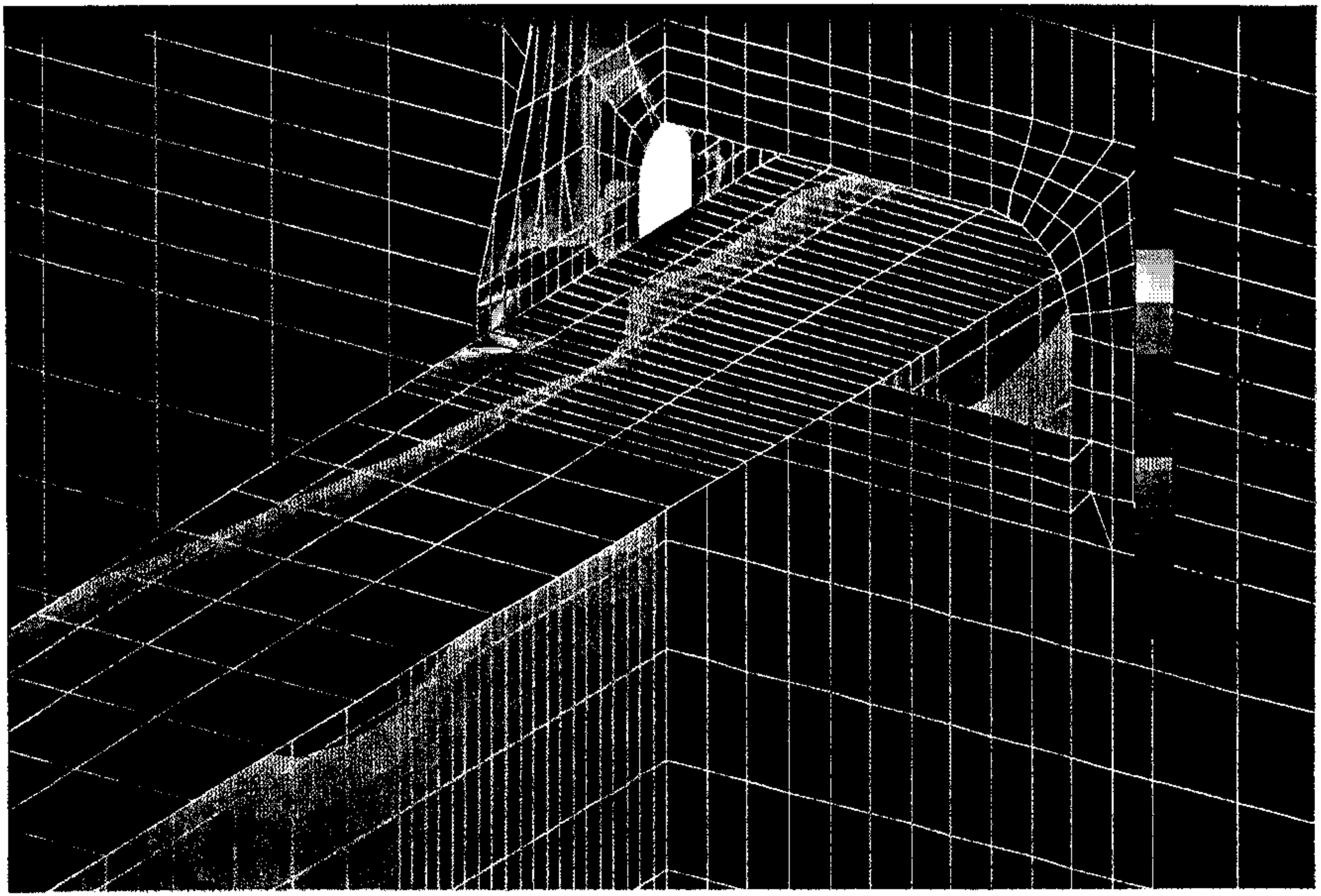



Lin STRESS Lc=1 A011 (A3)
Von Mises
2.00E+03
1.75E+03
1.50E+03
1.25E+03
1.00E+03





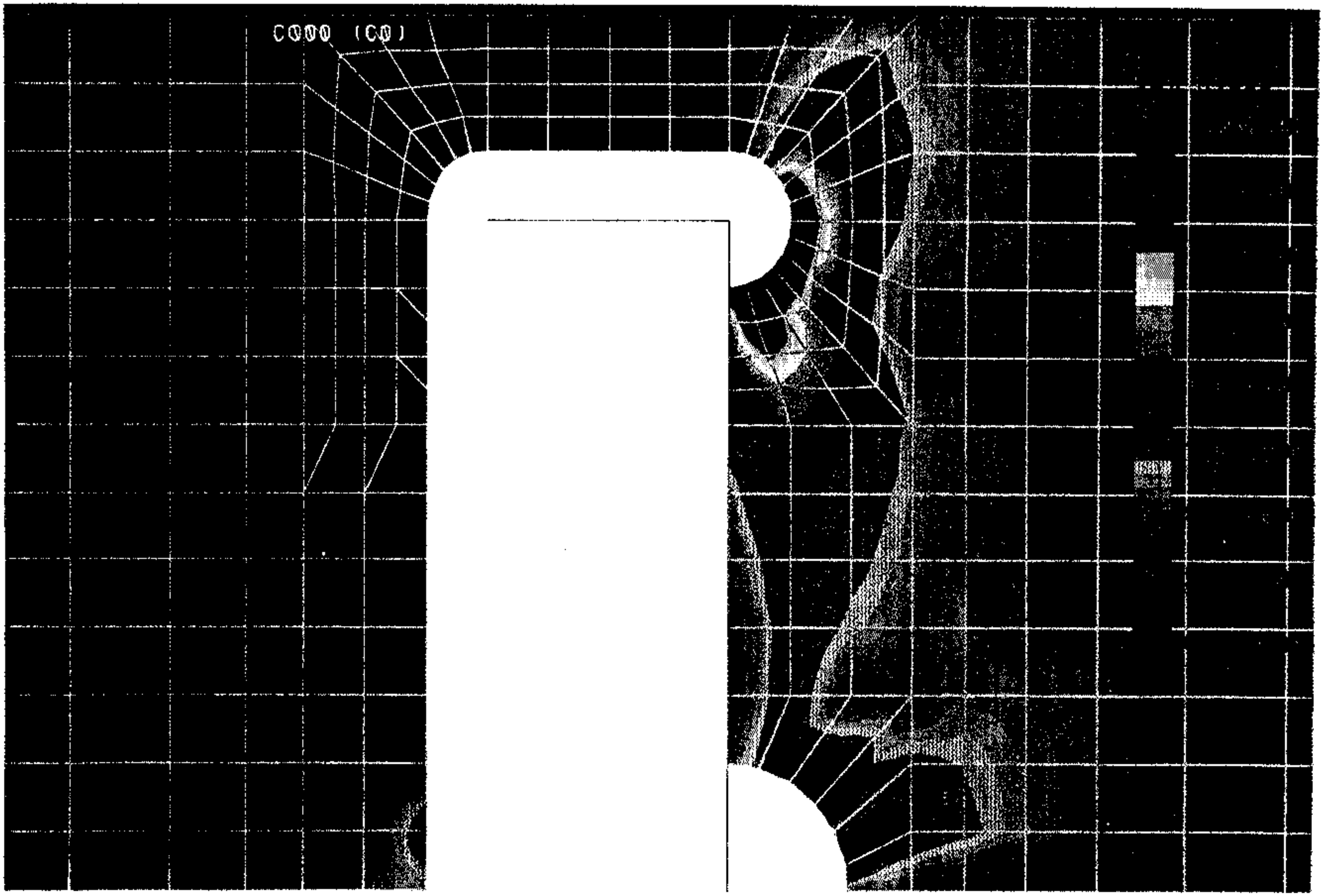


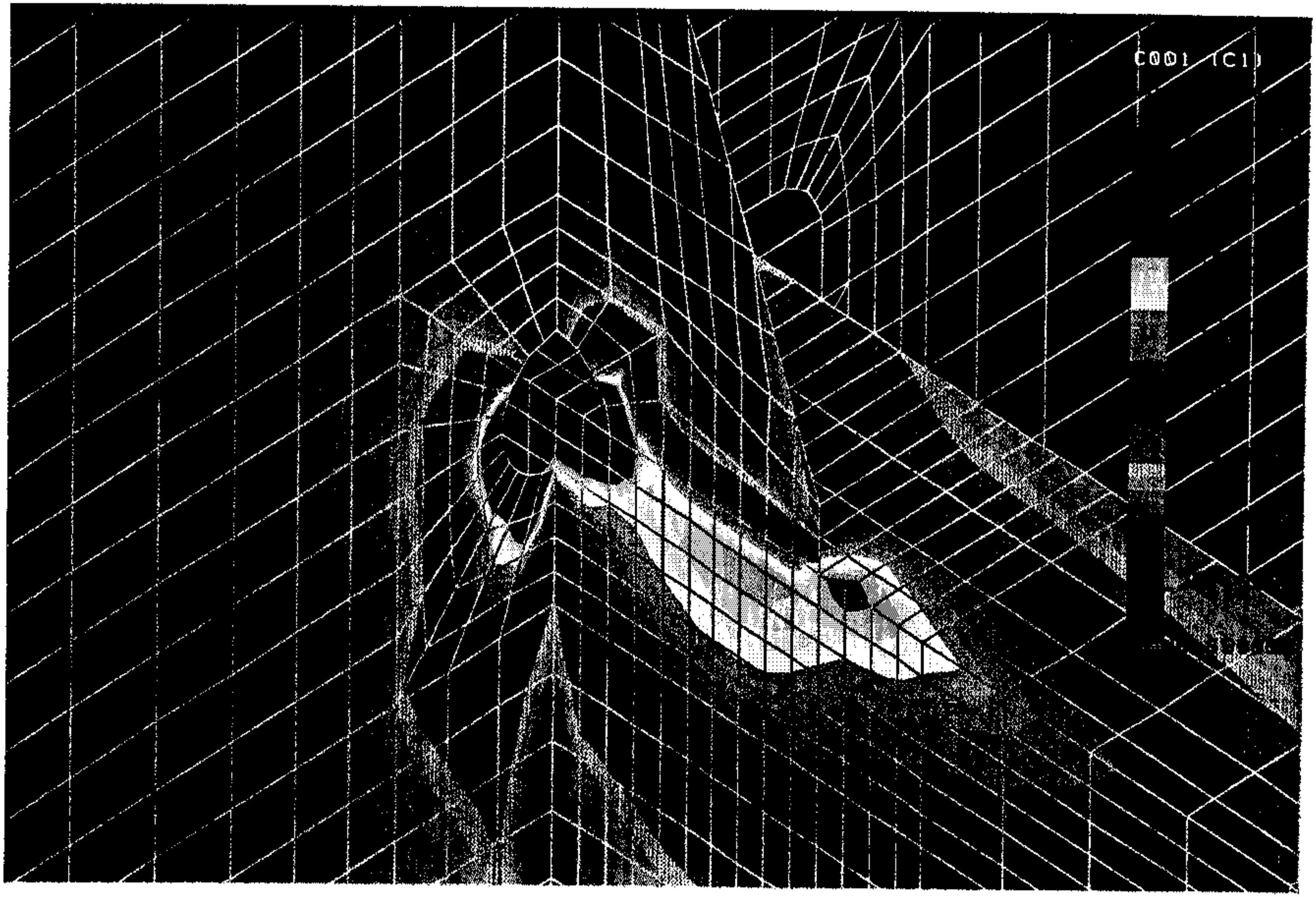


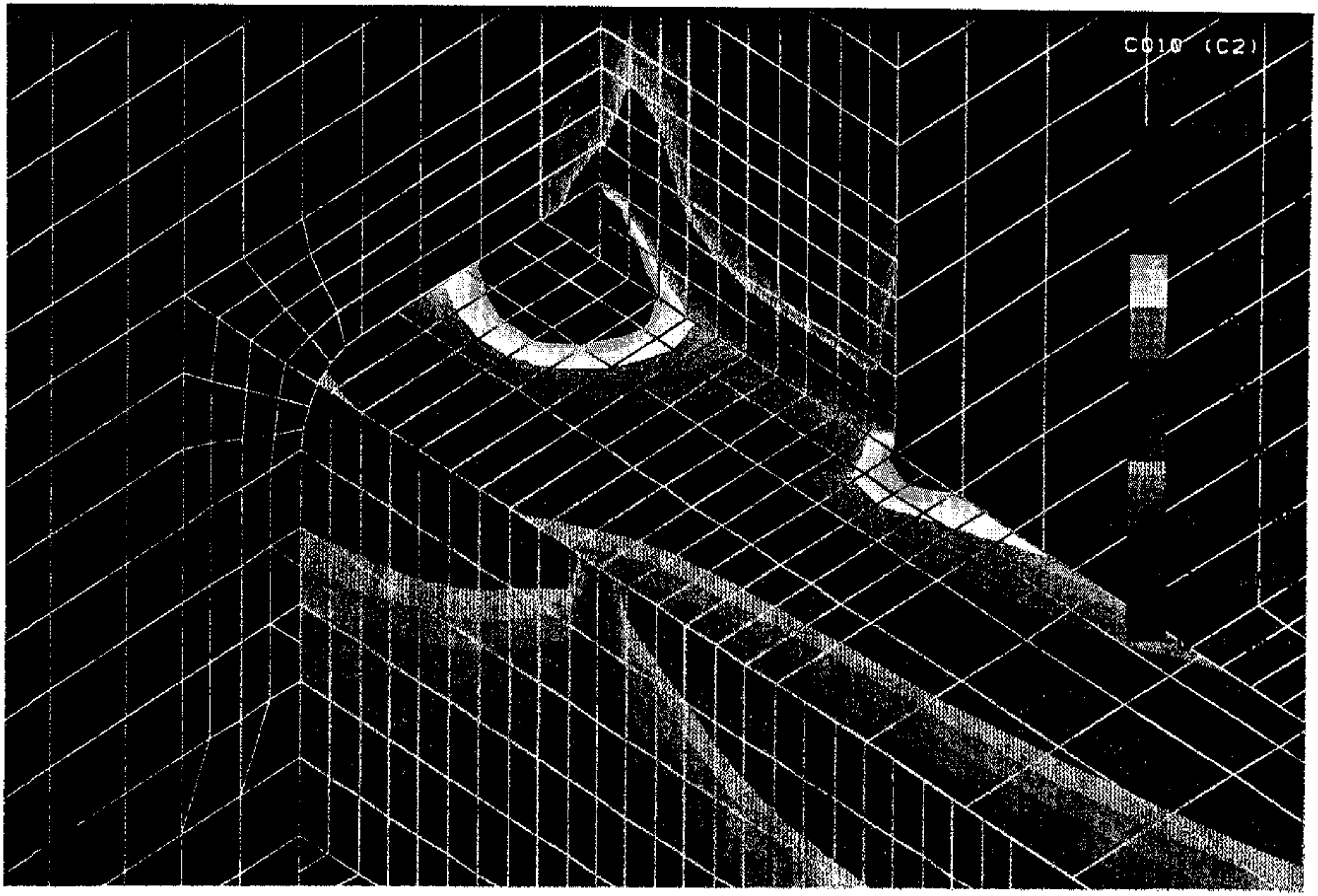
B.2 CSD #2 (model Cxxx)

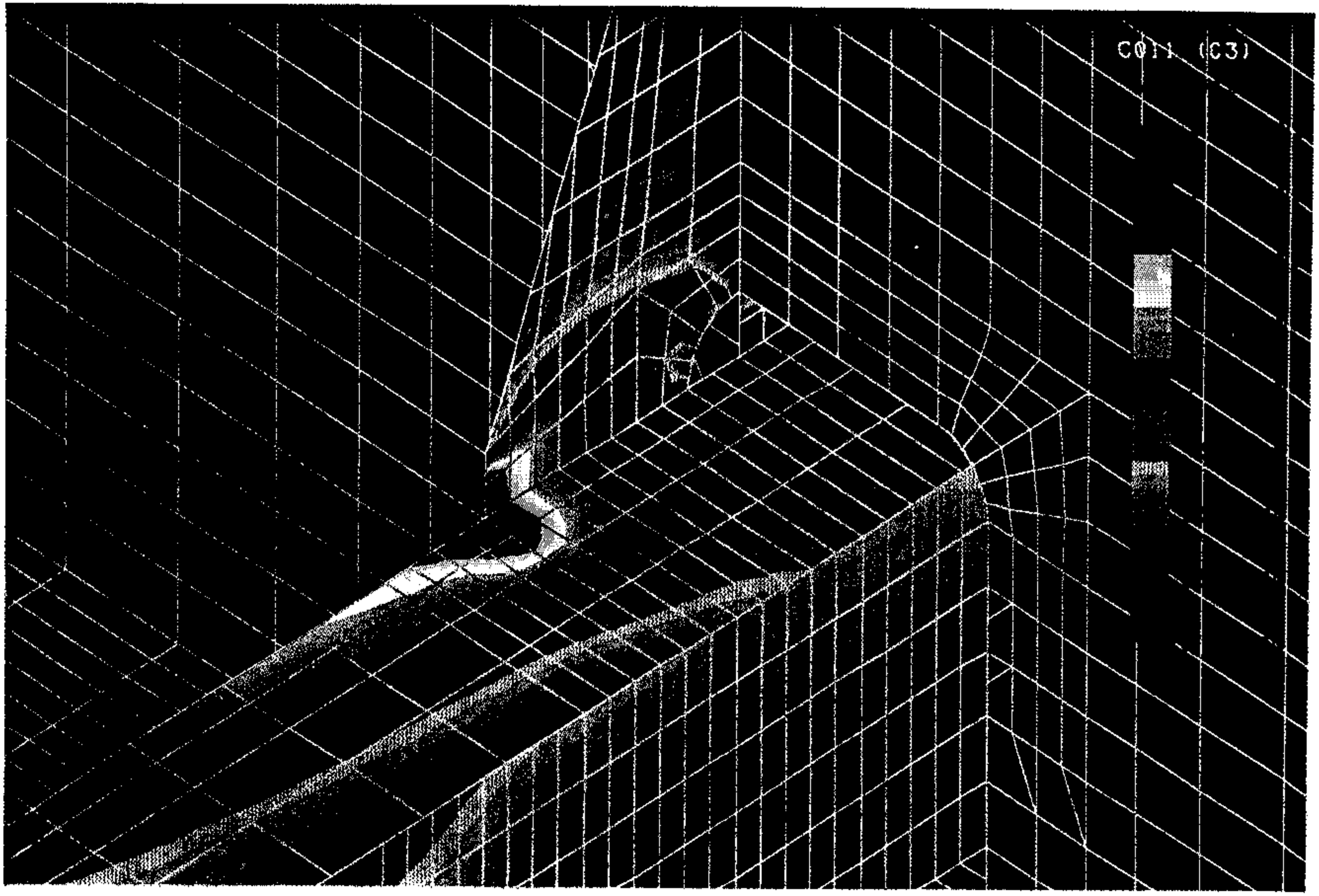
The following pages show the model meshing and the stress contours for the 8 models based on CSD #1. The 8 models are:

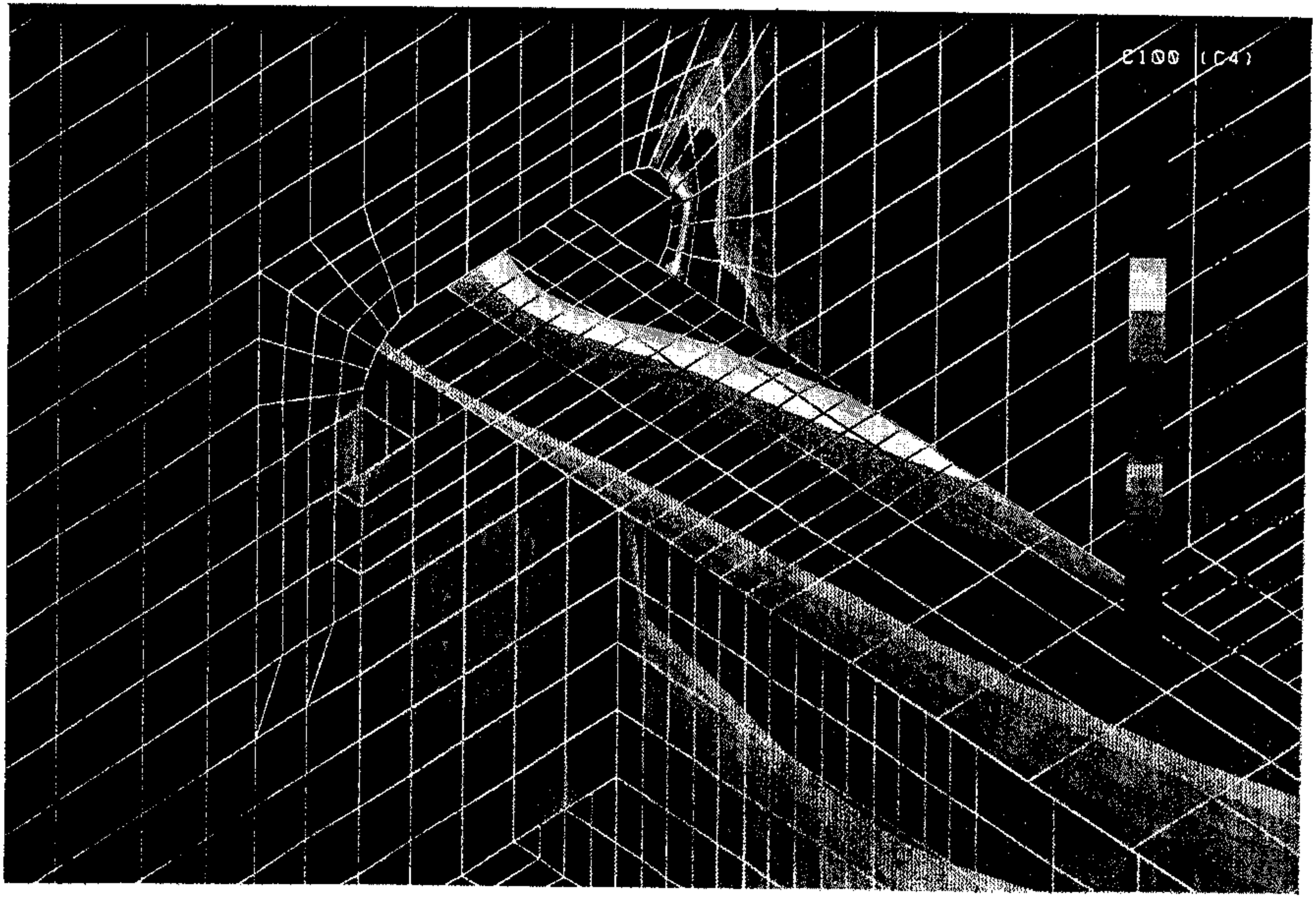
- C000 (C0) is a basic CSD.
- C001 (C1) is a basic CSD with bracket.
- C010 (C2) is a basic CSD with flatbar.
- C011 (C3) is a basic CSD with flatbar and bracket.
- C100 (C4) is a basic CSD with lug.
- C101 (C5) is a basic CSD with lug and bracket.
- C110 (C6) is a basic CSD with lug and flatbar.
- C111 (C7) is a basic CSD with lug, flatbar and bracket.

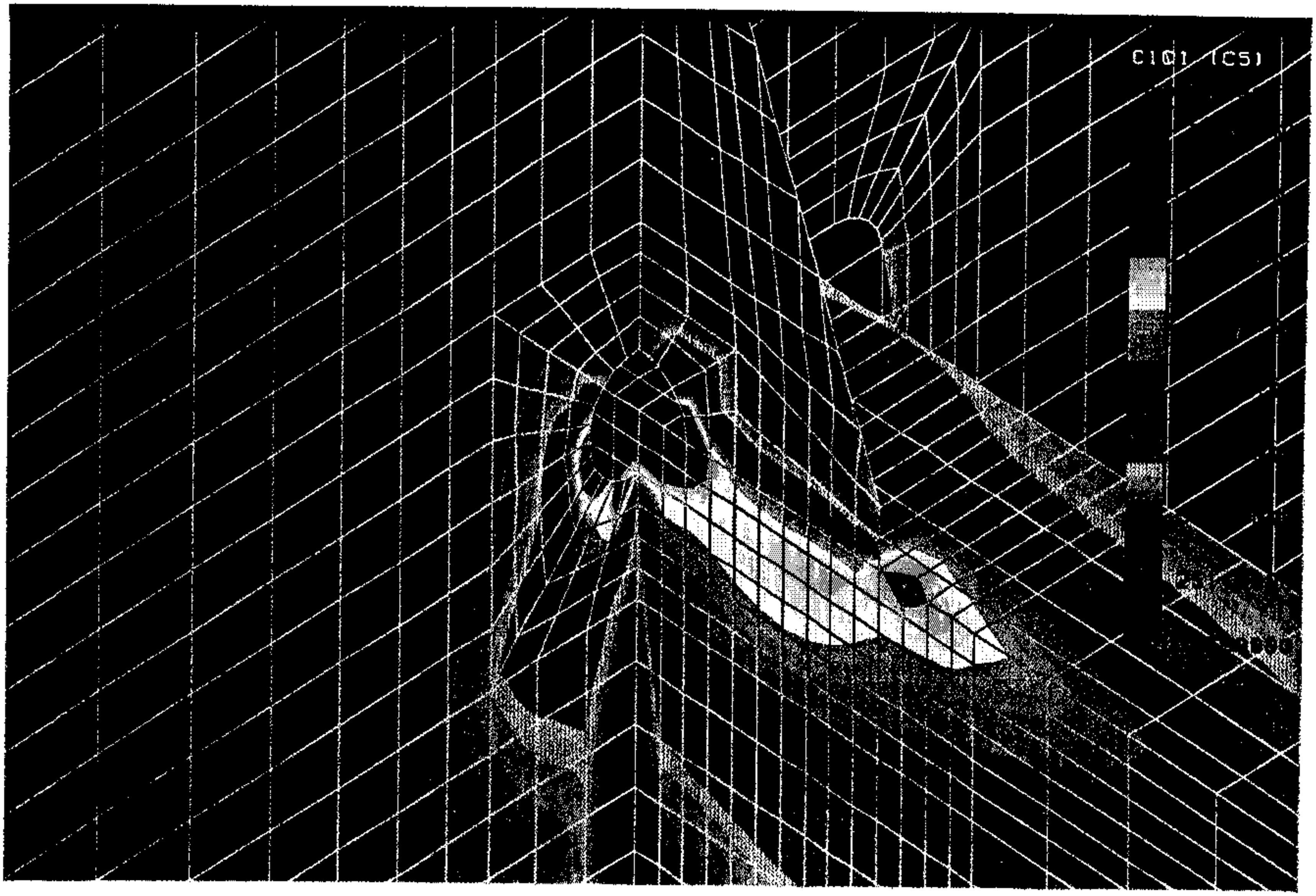

C000 (C0)

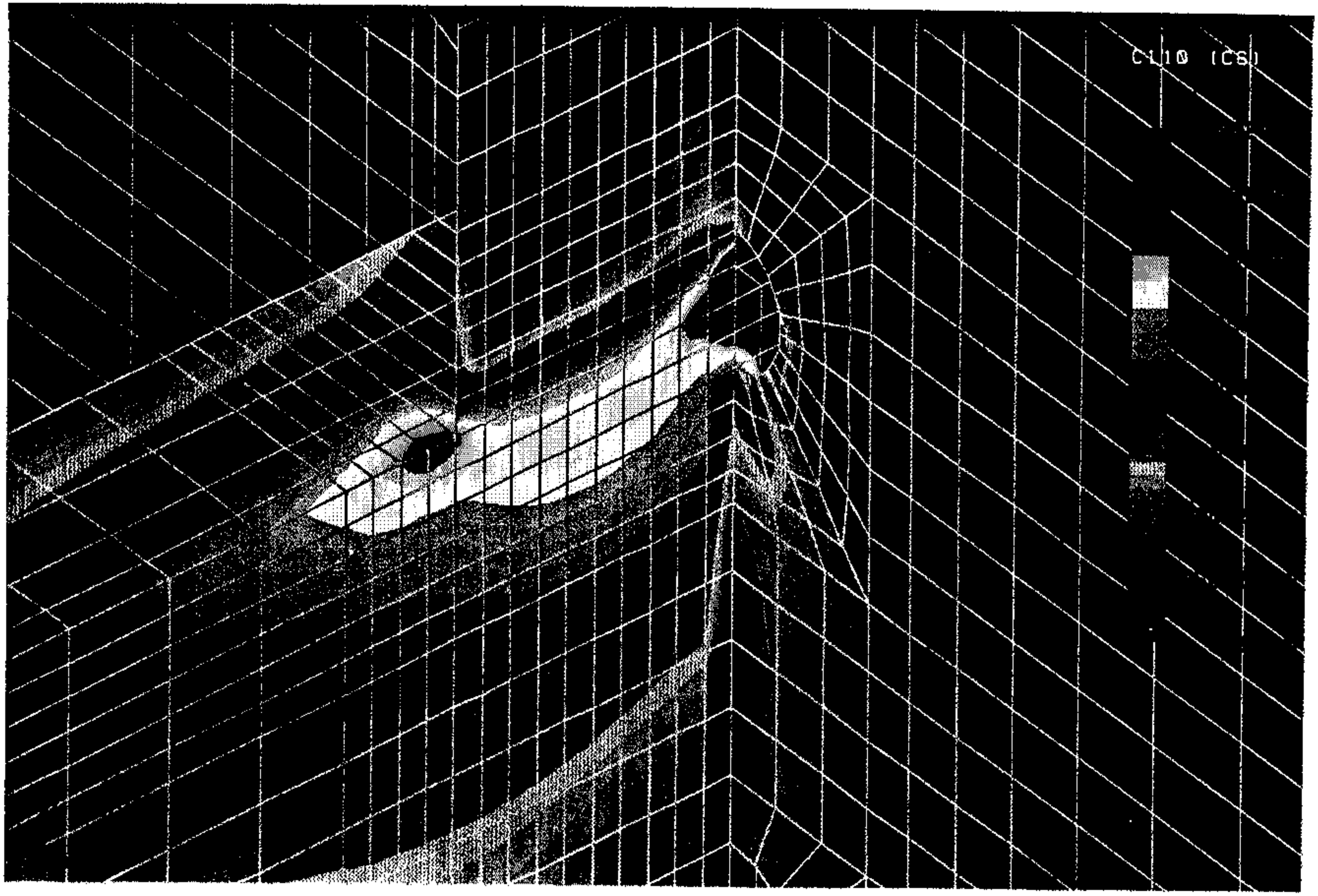

C001 (C1)

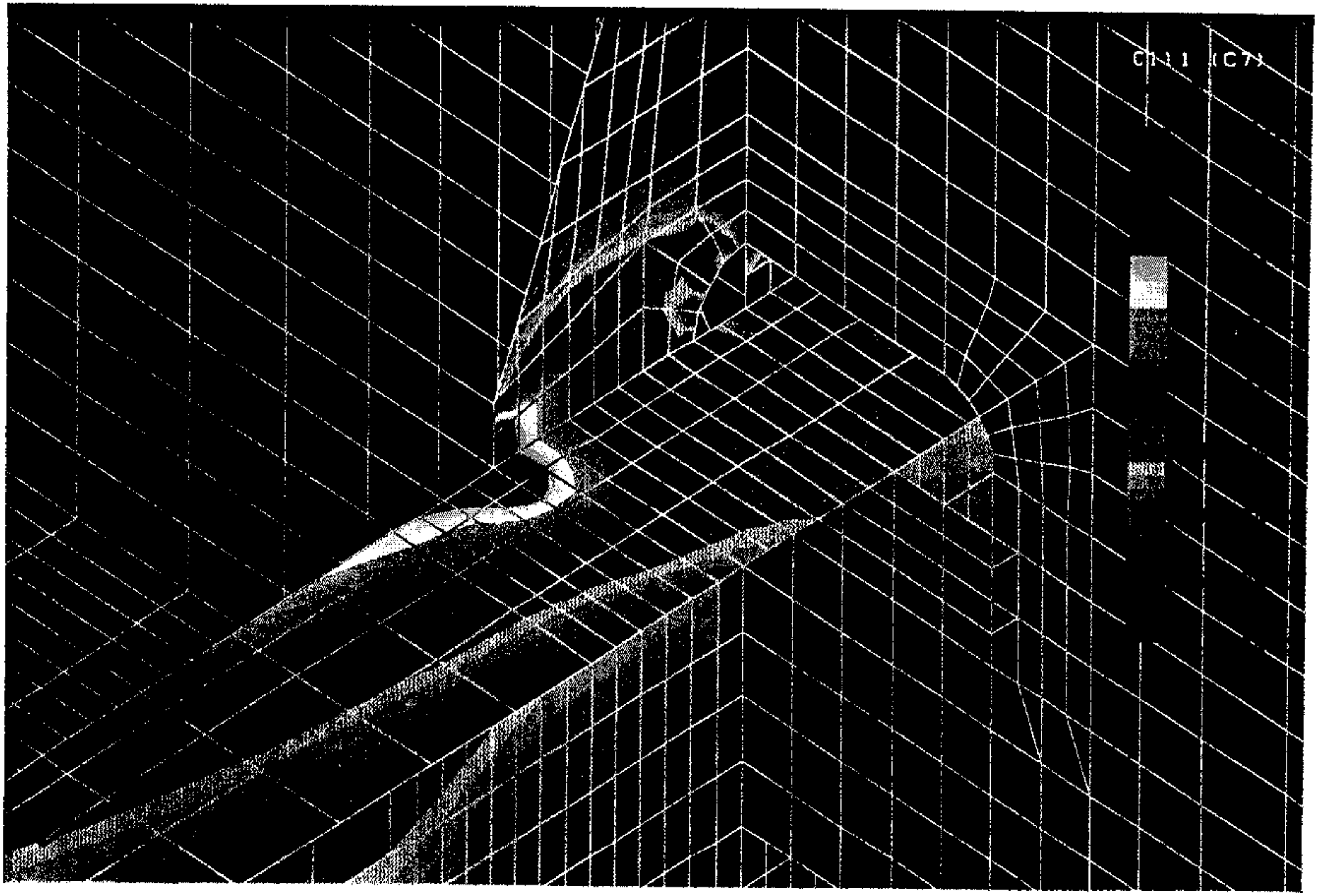

C010 (C2)


C011 (C3)


C100 (C4)


C1O1 (C5)

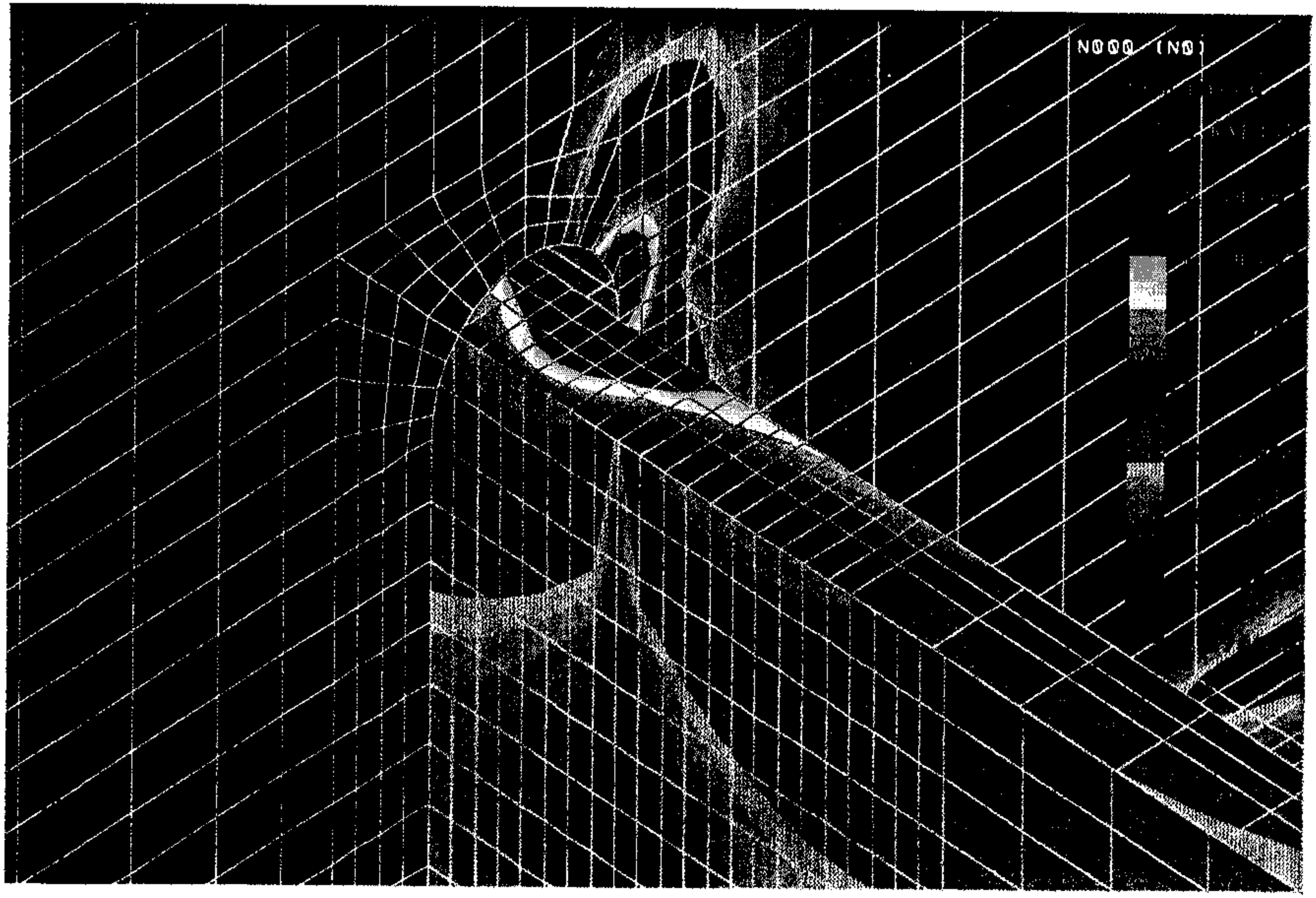

C110 (CS)

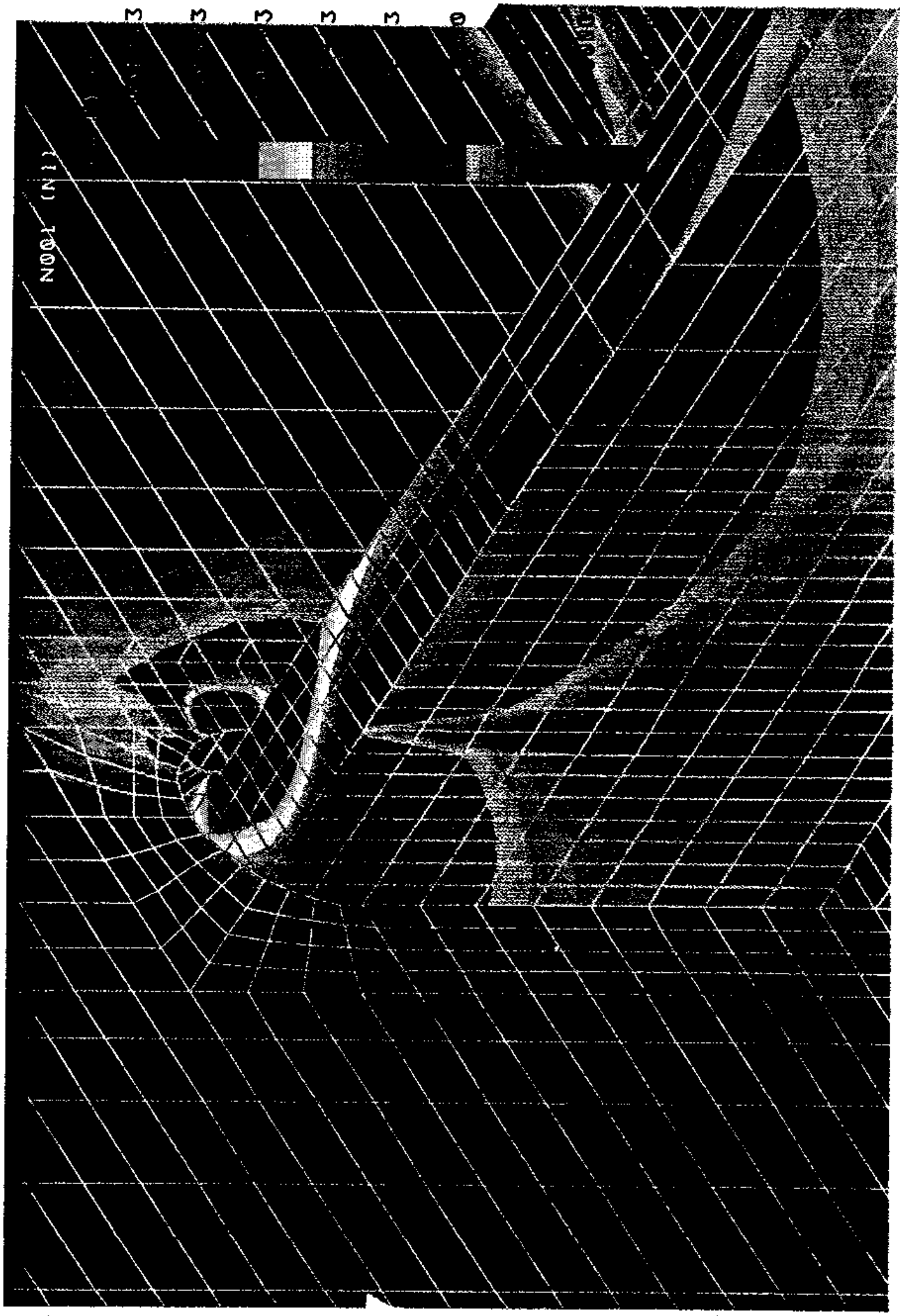

C111 (C7)

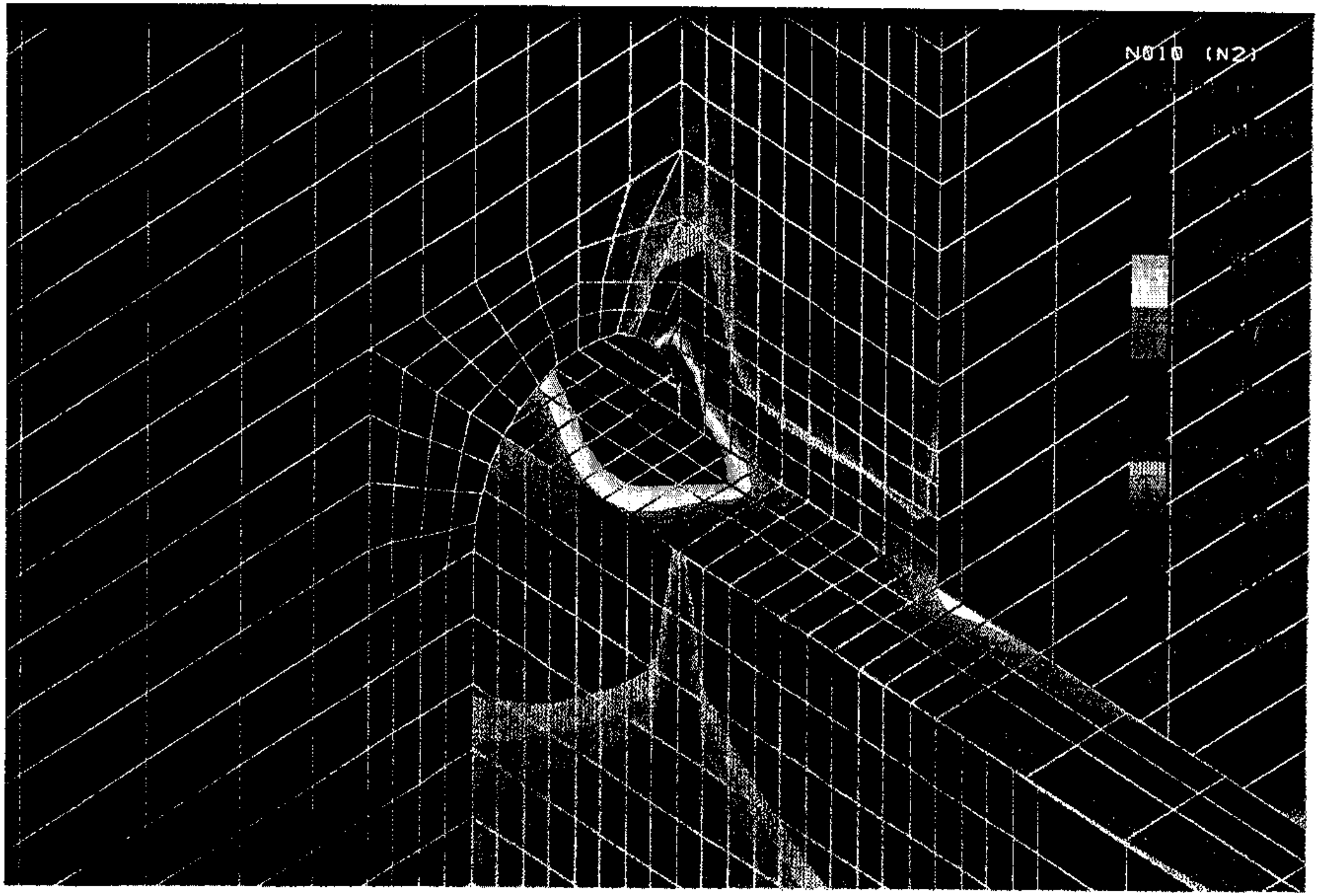
B.3 CSD #3 (model Nxxx)

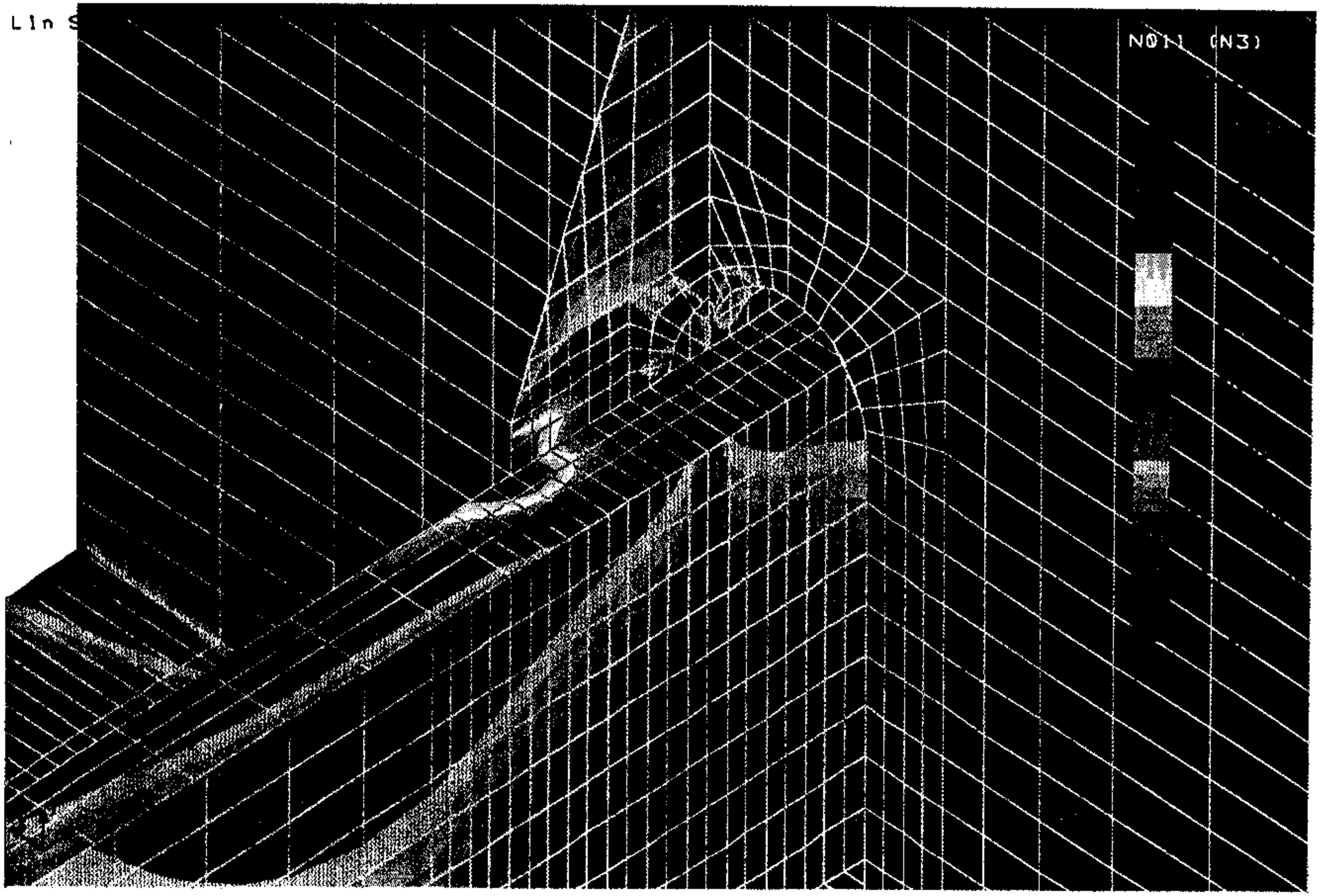
The following pages show the model meshing and the stress contours for the 8 models based on CSD #2. The 8 models are:

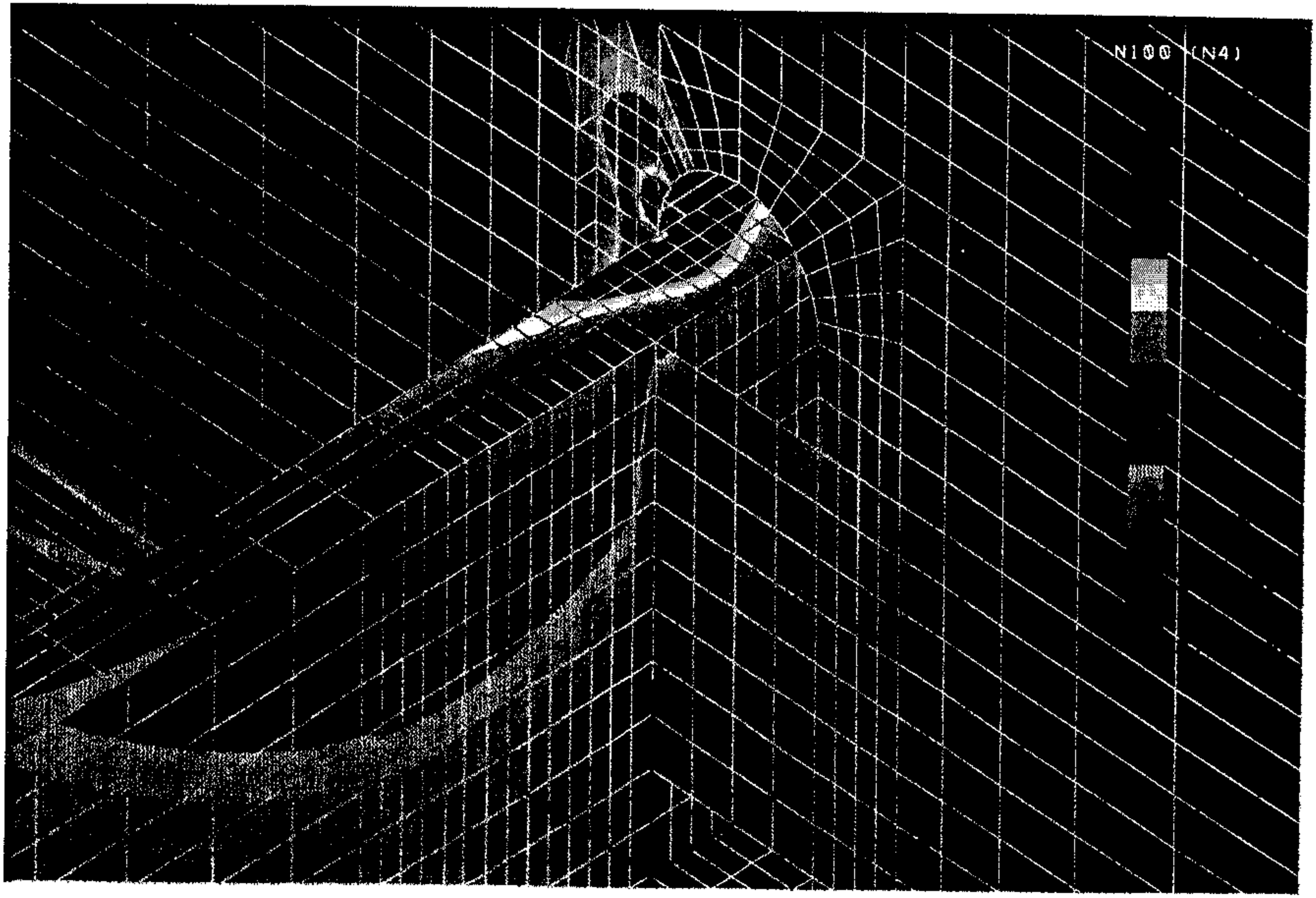
- N000 (N0) is a basic CSD.
- N001 (N1) is a basic CSD with bracket.
- N010 (N2) is a basic CSD with flatbar.
- N011 (N3) is a basic CSD with flatbar and bracket.
- N100 (N4) is a basic CSD with lug.
- N101 (N5) is a basic CSD with lug and bracket.
- N110 (N6) is a basic CSD with lug and flatbar.
- N111 (N7) is a basic CSD with lug, flatbar and bracket.

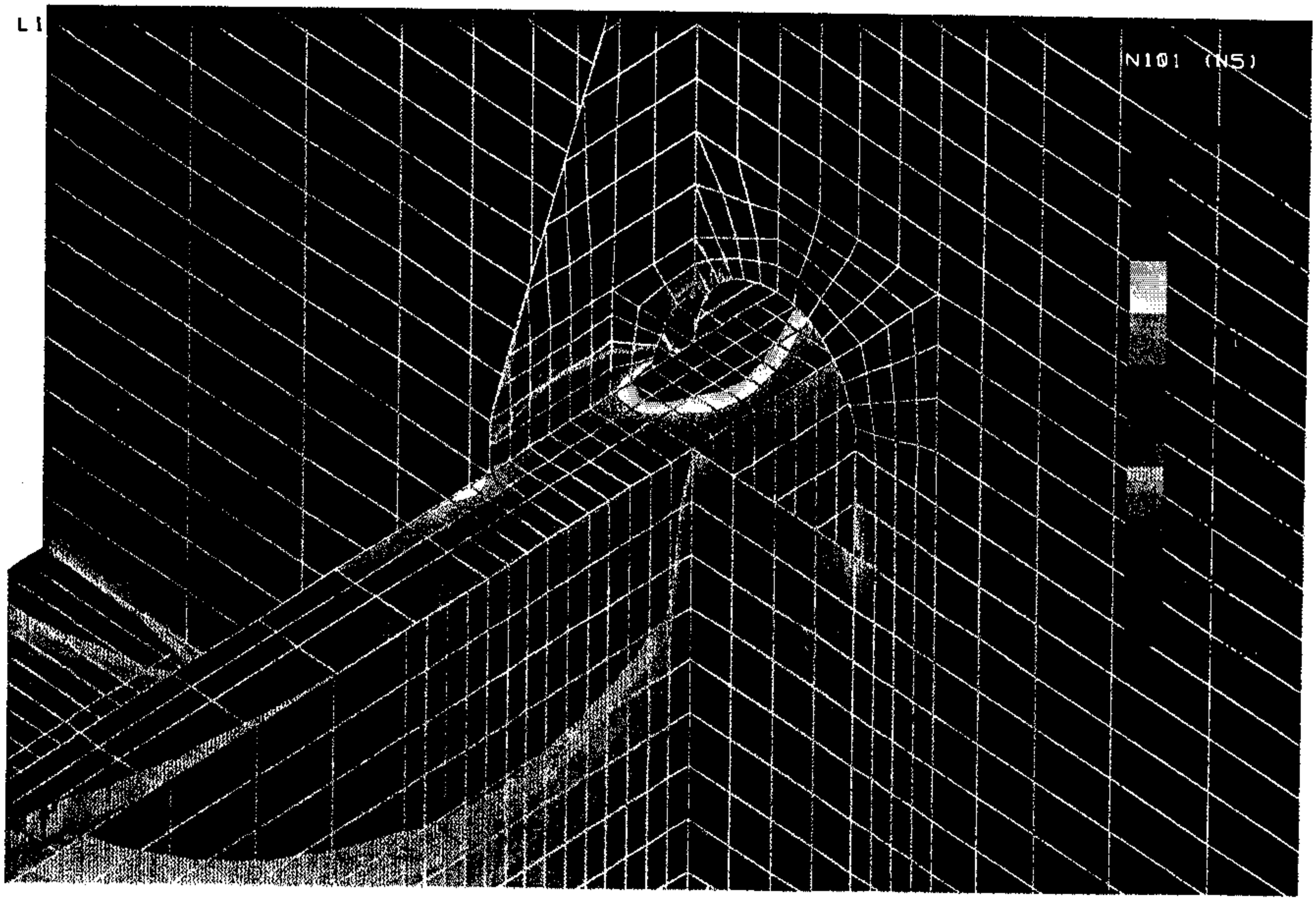

N000 (N0)

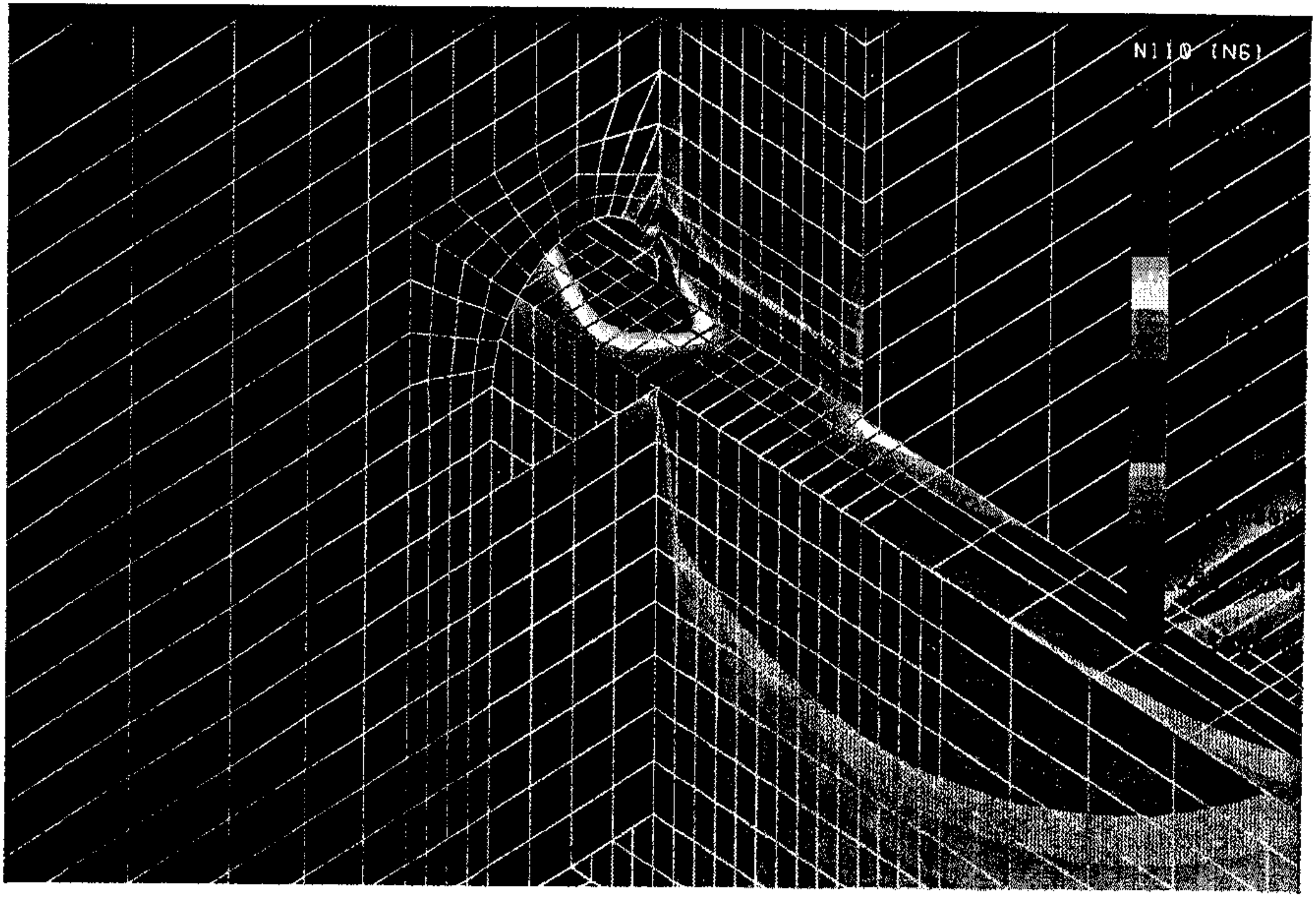


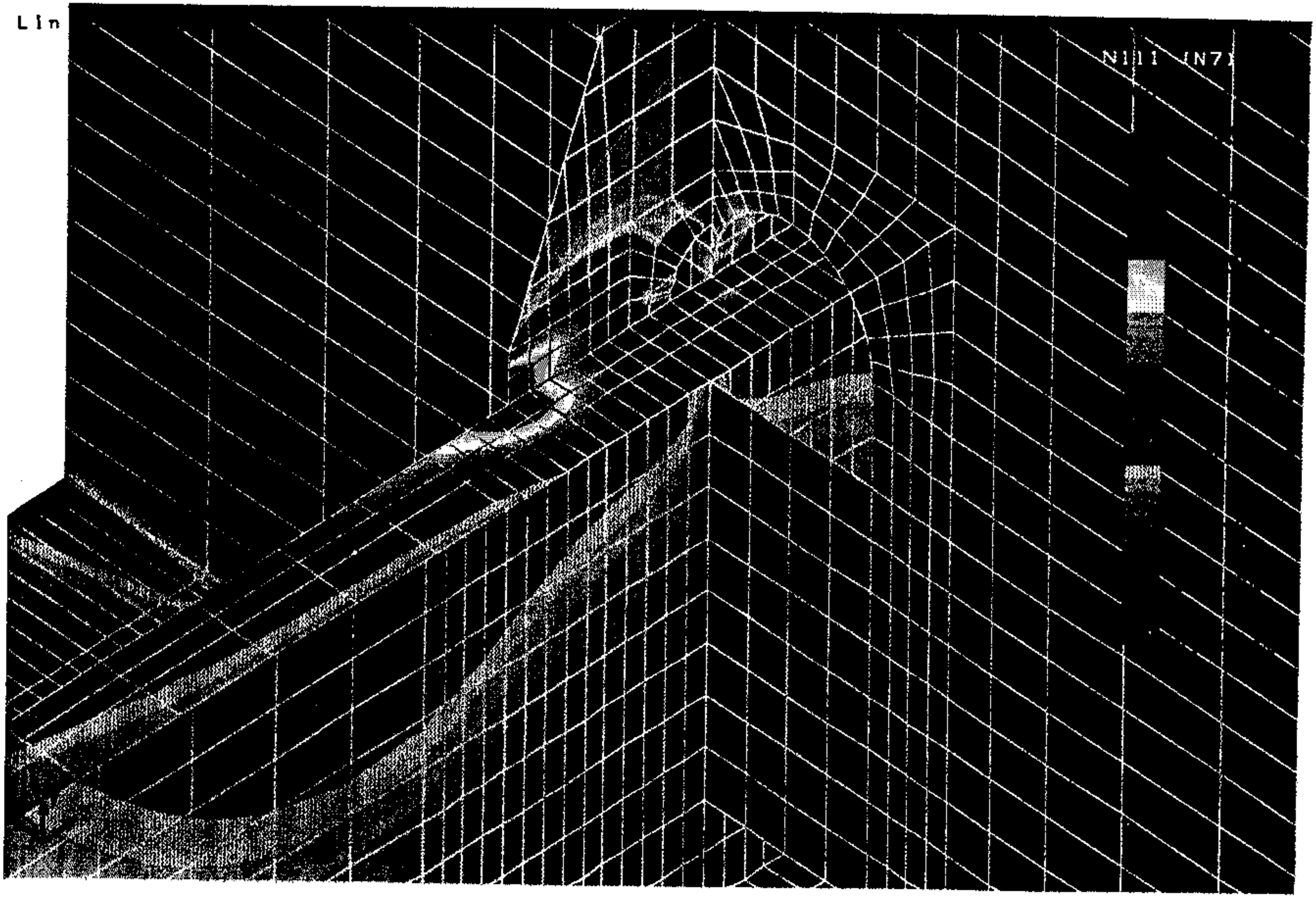

N010 (N2)


Lin S
N011 (N3)


N100 (N4)


L1
N101 (N5)


N110 (N6)


Lin
N111 (N7)

PROJECT TECHNICAL COMMITTEE MEMBERS

The following persons were members of the committee that represented the Ship Structure Committee to the Contractor as resident subject matter experts. As such they performed technical review of the initial proposals to select the contractor, advised the contractor in cognizant matters pertaining to the contract of which the agencies were aware, performed technical review of the work in progress and edited the final report.

Chairman

Paul Cojeen	U. S. Coast Guard

Members

LT Robert Holzman	U. S. Coast Guard
Fred Seibold	Maritime Administration
Dr. Walter MacLean	U. S. Merchant Marine Academy
Chao Lin	Maritime Administration
Dr. Y-k Chen	American Bureau of Shipping
William Siekierka	Naval Sea Systems Command, Contracting Officer's Technical Representative
Dr. Robert Sielski	National Academy of Science, Marine Board Liaison
CDR Steve Sharpe	U.S. Coast Guard, Executive Director Ship Structure Committee

351
352X

COMMITTEE ON MARINE STRUCTURES

Commission on Engineering and Technical Systems

National Academy of Sciences - National Research Council

The **COMMITTEE ON MARINE STRUCTURES** has technical cognizance over the interagency Ship Structure Committee's research program.

Dr. John Landes, *Chairman*, University of Tennessee, Knoxville, TN

Mr. Howard M. Bunch, University of Michigan, Ann Arbor, MI

Dr. Dale G. Karr, University of Michigan, Ann Arbor, MI

Mr. Andrew Kendrick, NKF Services, Montreal, Quebec

Dr. John Niedzwecki, Texas A & M University, College Station, TX

Dr. Alan Pense, NAE, Lehigh University, Bethlehem, PA

Dr. Barbara A. Shaw, Pennsylvania State University, University Park, PA

Dr. Robert Sielski, National Research Council, Washington, DC

CDR Stephen E. Sharpe, Ship Structure Committee, Washington, DC

DESIGN WORK GROUP

Dr. John Niedzwecki, *Chairman*, Texas A&M University, College Station, TX

Dr. Bilal Ayyub, University of Maryland, College Park, MD

Mr. Ovide J. Davis, Pascagoula, MS

Mr. Andy Davidson, NASSCO, San Diego, CA

Dr. Maria Celia Ximenes, Chevron Shipping Co., San Francisco, CA

Mr. Jeffrey Geiger, Bath Iron Works, Bath, ME

Mr. Hugh Rynn, Sea-Land Services, Elizabeth, NJ

MATERIALS WORK GROUP

Dr. Barbara A. Shaw, *Chairman*, Pennsylvania State University, University Park, PA

Dr. David P. Edmonds, Edison Welding Institute, Columbus, OH

Dr. John F. McIntyre, Advanced Polymer Sciences, Avon, OH

Dr. Harold S. Reemsnyder, Bethlehem Steel Corp., Bethlehem, PA

Dr. Bruce R. Somers, Lehigh University, Bethlehem, PA

RECENT SHIP STRUCTURE COMMITTEE PUBLICATIONS

Ship Structure Committee Publications - A Special Bibliography This bibliography of SSC reports may be downloaded from the internet at: "http://www.dot.gov/dotinfo/uscg/hq/nmc/nmc/ssc1/index.htm".

SSC-394 Strength Assessment of Pitted Plate Panels J. Daidola, J. Parente, I. Orisamolu, K-t. Ma 1997

SSC-393 Evaluation of Ductile Fracture Models R. Dexter, M. Gentilcore 1997

SSC-392 Probability Based Ship Design: Implementation of Design Guidelines A. Mansour, P. Wirsching, G. White, B. Ayyub 1996

SSC-391 Evaluation of Marine Structures Education in North America R. Yagle 1996

SSC-390 Corrosion Control of Inter-hull Structures M. Kikuta, M. Shimko, D Ciscom 1996

SSC-389 Inspection of Marine Structures L. Demsetz, R. Cario, R. Schulte-Strathaus, B. Bea 1996

SSC-388 Ship Structural Integrity Information System-Phase II M. Dry, R. Schulte-Strathaus, B. Bea 1996

SSC-387 Guideline for Evaluation of Finite Elements and Results R. I. Basu, K. J. Kirkhope, J. Srinivasan 1996

SSC-386 Ship's Maintenance Project R. Bea, E. Cramer, R. Schulte-Strauthaus, R. Mayoss, K. Gallion, K. Ma, R. Holzman, L. Demsetz 1995

SSC-385 Hydrodynamic Impact on Displacement Ship Hulls - An Assessment of the State of the Art J. Daidola, V. Mishkevich 1995

SSC-384 Post-Yield Strength of Icebreaking Ship Structural Members C. DesRochers, J. Crocker, R. Kumar, D. Brennan, B. Dick, S. Lantos 1995

SSC-383 Optimum Weld-Metal Strength for High Strength Steel Structures R. Dexter and M. Ferrell 1995

SSC-382 Reexamination of Design Criteria for Stiffened Plate Panels by D. Ghose and N. Nappi 1995

SSC-381 Residual Strength of Damaged Marine Structures by C. Wiernicki, D.