TR3314J

PROPOSITIONS

belonging to the Ph.D. thesis

Case-Based Reasoning for NDT Data Interpretation

by

Jacek Jarmulak

1. Whilst searching for suitable automatic knowledge-acquisition methods for CBR, one should not forget that much of the strength of CBR lies in the ability to combine a-priori domain knowledge and the data.
2. Re-use of complete AI systems, that is not only the tools but also the knowledge, must become a clear goal of AI. That is if one wishes the computer systems to reach human intelligence levels, and not only be good for "toy problem" solving.
3. Taking the development of various AI paradigms into consideration, one remains with the impression that the motto of AI scientists has been: let us build something that nobody really understands, and maybe intelligence will emerge.
4. As it is apparent that every few months computer hardware increases in speed and decreases in cost-factor, it would, in many situations, seem perfectly acceptable to sit back and wait with folded arms until the speed and storage related problems solve themselves.
5. Rather than being a medium for *free speech*, the Internet is a perfect tool for *total control*.
6. For a thesis to be considered complete, it should contain apart from the Acknowledgements also a Criticism section.
7. The judiciary seem to be of the opinion that "good" laws should at least be slightly contra to common sense.
8. To be free, is to able to voice that true is true and false is false. To say that true is false and false is true is no freedom but madness.
9. In modern society, people talk about human rights continually and, at the same time, they seem to become less and less responsible.
10. Though it is a bit strange when the worship in a church is limited to *(Hallelujah) x 16* kind of songs, it is really bad when this becomes their theology.
11. Civil servants should be replaced by computers, as all they do is follow rules. Even the poorest computer interface can be more user-friendly than certain civil servants.

STELLINGEN

behorende bij het proefschrift

Case-Based Reasoning for NDT Data Interpretation

door

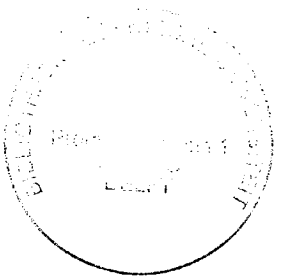
Jacek Jarmulak

1. Bij het zoeken naar methoden voor automatische kennisacquisitie voor CBR, moet men niet vergeten dat juist de mogelijkheid om a-priori domein kennis met de data te combineren één van de sterktes van CBR is.
2. Indien men wil dat computersystemen het menselijke intelligentieniveau bereiken en niet alleen voor "speelgoedproblemen" inzetbaar zijn, dan moet de herbruikbaarheid van complete AI systemen, dus van zowel de kennis als van de tools, een duidelijk doel binnen het vakgebied van de AI worden.
3. Als men de ontwikkeling van de verschillende AI paradigma's beschouwt, krijgt men de indruk dat het motto van de AI-wetenschappers als volgt luidt: laten we iets bouwen dat niemand begrijpt dan zal het misschien intelligent zijn.
4. Omdat computer hardware elke paar maanden een factor sneller en goedkoper wordt, is het in vele situaties zinvol om met de armen over elkaar te gaan zitten wachten totdat het snelheidsprobleem zich vanzelf oplost.
5. Internet is eerder een perfect middel voor *total control* dan een medium voor *free speech*.
6. Om echt compleet te zijn zou elk proefschrift behalve een Dankwoord ook een Verwijtwoord moeten bevatten.
7. Wetgevers lijken te denken dat "goede" wetten ten minste een beetje tegen het gezonde verstand in moeten gaan.
8. Vrij zijn is te mogen zeggen dat waar waar en dat fout fout is. Te zeggen dat waar fout en fout waar is, is geen vrijheid maar dwaasheid.
9. Des te meer de mensen over de mensenrechten praten in de moderne maatschappij, des te minder verantwoordelijk lijken ze te zijn.
10. Al is het een beetje vreemd dat de lofprijzing in een kerk beperkt is tot *(Alleluja) x 16* soort liederen, het is pas echt een ramp als dit tot theologie verheven wordt.
11. Sommige ambtenaren zouden door computers vervangen moeten worden. Immers: al wat zij doen is regels opvolgen en bovendien kan zelfs een slechte computer interface gebruiksvriendelijker zijn dan menig ambtenaar.

Case-Based Reasoning for NDT Data Interpretation

The cover shows a schema of CBR interpretation cycle placed over a background of complex-valued signals from eddy-current inspection of a heat exchanger. The tubes in the heat exchanger are made of thin-walled welded austenic steel. The periodic big indications are from baffles. The "noise" is caused by the weld. Apart from that, the signal contains indications from ferritic inclusions and from at least five defects — can you spot them?.

Case-Based Reasoning for NDT Data Interpretation



Proefschrift

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus prof. ir. K.F. Wakker,
in het openbaar te verdedigen ten overstaan van een commissie,
door het College voor Promoties aangewezen,
op dindag 6 april 1999 te 13.30 uur

door

Jacek JARMULAK

informatica ingenieur
geboren te Zgorzelec, Polen

Dit proefschrift is goedgekeurd door de promotor:
Prof. dr. H. Koppelaar

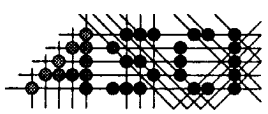
Samenstelling promotiecommisie:

Rector Magnificus	Technische Universiteit Delft, voorzitter
Prof. dr. H. Koppelaar	Technische Universiteit Delft, promotor
Dr. ir. E.J.H. Kerckhoffs	Technische Universiteit Delft, toegevoegd promotor
Prof. dr. ir. E. Backer	Technische Universiteit Delft,
Prof. dr. M. Donze	Technische Universiteit Delft,
Prof. dr. F.C.A. Groen	Universiteit van Amsterdam,
Prof. ir. H.B. Verbruggen	Technische Universiteit Delft,
Dr. habil. K.-D. Althoff	Fraunhofer Institute for Experimental Software Engineering

Published and distributed by:
Jacek Jarmulak
e-mail: jacek@scms.rgu.ac.uk
jacek@kgs.twi.tudelft.nl

ISBN 0 9535094 0 0

Printed in United Kingdom



This work was carried out in the Advanced School for Computing and Imaging (ASCI).
ASCI dissertation series number 37.

Preface

The research, the results of which are described in this thesis, has been done within the ANDES (Automatic NonDestructive Evaluation System) project. The work has been done at TNO Institute of Applied Physics, and has been sponsored by Shell, Akzo-Nobel, NS Ultrasoonbedrijf, and Materiaal Metingen Testgroep BV. The goal of the ANDES project has been to use Artificial Intelligence techniques for the interpretation of data coming from nondestructive testing (NDT). Two practical data interpretation problems were to be solved: interpretation of eddy-current data from heat-exchanger inspection (this problem was proposed by Shell and Akzo-Nobel) and interpretation of ultrasonic B-scan images from rail inspection (proposed by NS Ultrasoonbedrijf).

The fact that there were two problems to be solved had both its advantages and disadvantages. A disadvantage is that because of the lack of time it was only possible to develop prototype systems for each problem. Concentrating on just one problem could have made it possible to develop a system suitable for real field inspections. Also, the techniques used could have been subjected to a more thorough theoretical analysis and evaluation.

The advantage is that having to work on two problems which are quite different from each other gives more insight in the problem of automating NDT data interpretation. Based on these two problems, additional study of literature, and discussions with people from the NDT field, one may make some reasonably well founded generalisations about automating NDT data interpretation, in particular using AI techniques.

The chapters of this thesis are, to a large extent, self-contained (for example, each contains a list of references mentioned in the chapter), therefore, they can be read separately or some chapters can be omitted while reading. Of course, because of the wide range of topics discussed in this thesis it is impossible to be complete, however, it has been attempted to give a good overview of NDT and of AI data interpretation techniques. Where choices had to be made, topics were selected which are related to the two test cases that were the subject of study.

I would like to thank my supervisors during the project: Peter Paul van't Veen from TNO-TPD and Eugene J.H. Kerckhoffs from TU Delft. Thanks also to colleagues at TNO-TPD and to prof. Henk Koppelaar whose comments have helped to make this thesis better.

Readers new to NDT should note also other terms are commonly used to name what is essentially the same field. Thus one may also use NDI (nondestructive inspection) or NDE (which may mean either nondestructive evaluation or examination). Out of these terms the nondestructive evaluation seems to cover the widest scope. In this thesis it has been attempted to consistently use NDT.

> *"It often does more harm than good to force definitions on things we don't understand. Besides, only in logic and mathematics do definitions ever capture concepts perfectly. The things we deal with in practical life are usually too complicated to be represented by neat, compact expressions. Especially when it comes to understanding minds, we still know so little that we can't be sure our ideas about psychology are even aimed in the right directions. In any case, one must not mistake defining things for knowing what they are."*
>
> *— Marvin Minsky, from "The Society of Mind", 1985.*

Summary

of the thesis

"Case-Based Reasoning for NDT Data Interpretation"

by Jacek Jarmulak

Nondestructive testing (NDT) is a name for a range of methods and procedures used to determine fitness of industrial products for further use. Testing is necessary because such products may contain various defects either being the results of faults or inaccuracies in the manufacturing process or the result of use. Some of these defects can be easy to spot (certainly if they have already led to a failure) but some are not immediately visible. Their presence can be determined, without negatively affecting the inspected object, using special (NDT) techniques.

Many techniques, based on various physical phenomena, are used in NDT. Some of the most popular techniques make use of eddy currents, ultrasound, X-rays, gamma-rays, and acoustic emissions. An NDT system contains some energy source which emits energy that interacts with the inspected object. The interaction is registered using special sensors. The registered signal has then to be analysed to determine if it contains any indications of possible defects. The interpretation of data is often difficult and requires special training and expertise.

Because of increasing safety requirements, and because of the desire to extend the effective lifetime of products, an increasing amount of NDT inspection is being done. Ongoing mechanisation and automatisation of nondestructive testing procedures lead to an increasing amount of data acquired. This, in turn, creates a need for reliable automatic or automated data-interpretation techniques to reduce the workload on the operators.

Though the use of automatic interpretation techniques serves, first of all, the increase in the speed of data interpretation, the increase in the reliability, precision, and reproducibility of the inspection are also important goals. Moreover, because computer systems are capable of analysing more data simultaneously than could be done by an operator, they can handle and interpret more complex data than an operator could.

In response to these developments in NDT, the ANDES (Automatic NonDestructive Evaluation System) project was started at TNO. The goal of the project was to investigate possibilities for the automation of NDT data interpretation and to demonstrate effective and feasible methods on two representative NDT problems. The emphasis was on investigating usefulness of Artificial Intelligence (AI) techniques. The two test cases which guided the research were: interpretation of data coming from an ultrasonic rail-inspection train (owned and operated by NS Ultrasoonbedrijf) and interpretation of eddy-current data from heat-exchanger inspection.

Artificial Intelligence provides a set of techniques for handling complex problems, usually needing human intelligence to solve. The most popular techniques are: artificial neural networks (ANNs) trying to model neural organisation of the brain, and rule-based expert systems which use expert knowledge represented in form of rules to solve problems. A more recent AI technique is case-based reasoning (CBR). CBR is based on the observation that people usually solve problems by reusing solutions that have worked in the past. This works because usually similar problems have similar solutions.

A CBR system stores previously solved problems, so-called cases, in a case-base. When a new problem has to be solved a search is done in the case-base and the most similar previously solved problem is retrieved. Based on the old solution, a solution to the new problem is determined. Because usually the retrieved problem is not exactly the same as the new problem, some adaptation of the old solution is necessary to fit the new problem. Once the solution has been verified to be correct, it can be saved in the case-base. By saving new cases a CBR system is improving its performance — it learns. Ability to learn is only one of the advantages of CBR systems. Other advantages are: relative ease of acquiring knowledge necessary to build the system (as it is contained mainly in cases), high reliability due to use of cases as contextualised knowledge representations and the capability to recognize new situations, reduced need for maintenance as the system can adapt to many changes in the problem domain. If maintenance is necessary it is relatively easy, usually confined to the maintenance of the case-base.

The advantages of CBR seem to make it especially suited for NDT problems, where the knowledge about data interpretation is often difficult to formalise, the reliability is crucial, and ease of maintenance is important from the economic point of view. Because CBR systems learn they can adapt to changing inspection conditions. Moreover, the normal way of doing data interpretation by NDT inspectors is, to a large degree, CBR-like.

Within the ANDES project two prototype CBR applications have been developed, one for each of the already mentioned test cases. The tests conducted on data from real field measurements were satisfactory for both systems. The CBR system for URS is currently being prepared for use on the inspection train.

During the development of the two systems, the issues of optimal case representation and case matching demanded much attention, as they are crucial to the retrieval of the most similar cases of the already classified data from the case-base. Related to this is appropriate data preprocessing, like removing noise and other unwanted influences, which makes the data easier to interpret. Also, obtaining data necessary for testing of the system proved to be far from trivial. Moreover, the URS system demanded considerable effort in optimally designing case-base to achieve acceptable retrieval times.

Though the two developed prototypes, as well as other systems described in the NDT literature, demonstrate the capabilities of AI for NDT data interpretation, the use of such systems for real inspections is still almost nonexistent. One reason for this may be the complexity of the factors involved in the decision making about NDT inspection. Though, in principle, it is possible to demonstrate the advantages of using automated data interpretation systems (or even fully automated inspection systems) in clear economical terms, such analyses are complex, and make sense only for particular test cases (generally applicable analyses are not feasible). Moreover, introducing new techniques (not only for data interpretation) in NDT is made difficult by a common involvement of at least three parties in the inspection: the inspection-system manufacturer, the inspection company doing the actual inspections, and the owner of the inspected product or installation. Successful introduction of new systems requires cooperation of all the parties and it is not easy to achieve. Possibly, if automated data interpretation systems become cheaper and easier to construct and to use, without compromising reliability, then their introduction becomes easier. CBR is a technique which may help to achieve this.

Samenvatting

van het proefschrift
"Case-Based Reasoning for NDT Data Interpretation"
door Jacek Jarmulak

De aanduiding "Niet Destructief Onderzoek" (NDO) staat voor een aantal methoden en procedures welke gebruikt worden om de geschiktheid voor verder gebruik te bepalen van (hoofdzakelijk) industriële producten. Dit onderzoek is noodzakelijk omdat deze producten verschillende defecten kunnen bevatten als gevolg van fouten en onnauwkeurigheden bij de productie of als gevolg van het gebruik. Sommige defecten zijn eenvoudig te detecteren (zeker als ze al tot een schade geleid hebben), maar de meerderheid is niet direct zichtbaar. Door gebruik te maken van speciale (NDO) technieken kan de aanwezigheid van deze defecten worden geconstateerd zonder op nadelige wijze (niet destructief) het te onderzoeken object te beïnvloeden.

Binnen het NDO worden een verscheidenheid aan technieken gebruikt, gebaseerd op allerlei natuurkundige fenomenen. De meest populaire NDO technieken maken gebruik van wervelstromen, ultrageluid, Röntgenstralen, gammastralen, en akoestische emissie. Een NDO systeem beschikt over een energiebron welke energie in een bepaalde vorm uitzendt. De wisselwerking van de uitgezonden energie en het te onderzoeken object wordt geregistreerd door geschikte sensoren. Het geregistreerde signaal moet worden geanalyseerd om te bepalen of het indicaties van mogelijke defecten bevat. Deze interpretatie van de gegevens is vaak moeilijk en vereist opleiding en expertise.

Vanwege het strenger worden van de veiligheidseisen en ook vanwege de wens om de bruikbare leeftijd van producten te verlengen, worden er steeds meer NDO inspecties uitgevoerd. Verder resulteert verregaande mechanisatie en automatisering van niet-destructieve inspectie procedures in een stijgende hoeveelheid van geregistreerde data. Dit veroorzaakt een behoefte aan betrouwbare automatische of geautomatiseerde data interpretatie technieken die de werklast van de operators kunnen beperken. Behalve het verhogen van de snelheid van data-interpretatie zijn andere belangrijke doelen het verhogen van de betrouwbaarheid, de nauwkeurigheid en de reproduceerbaarheid van de inspectie. Het gebruik van computersystemen ligt hierbij voor de hand, omdat ze in staat zijn om grote hoeveelheden gegevens in korte tijd te analyseren, zodat ze in dezelfde tijd meer en complexere data kunnen interpreteren dan een operator.

Als antwoord op deze ontwikkelingen binnen het NDO is bij TNO het ANDES (Automatisch Niet-Destructief Evaluatie Systeem) project gestart. Het doel van het project was om de mogelijkheden voor het automatiseren van de data-interpretatie te onderzoeken en om, aan de hand van twee representatieve NDO problemen, doeltreffende en realiseerbare methoden te demonstreren. De nadruk lag op het onderzoeken van de geschiktheid van Artificiële Intelligentie (AI) technieken. Aan de basis van het onderzoek stonden twee casussen: interpretatie van data afkomstig van het Ultrasone Railinspectie Systeem (URS), operationeel op een trein van het NS Ultrasoonbedrijf, en interpretatie van wervelstroom data afkomstig van de inspectie van warmtewisselaars.

Het vakgebied van de Artificiële Intelligentie levert een aantal technieken om complexe problemen aan te pakken welke normaal gesproken alleen op te lossen zijn door gebruik te maken van menselijke intelligentie. De twee meest bekende technieken zijn: kunstmatige neurale

netwerken (KNN), welke de neurale organisatie van de hersens proberen na te bootsen, en regel-gebaseerde expert systemen, welke expert kennis gerepresenteerd in de vorm van regels gebruiken om problemen op te lossen. Een meer recente AI techniek is "case-based reasoning" (CBR). CBR is gebaseerd op de observatie dat mensen problemen oplossen door gebruik te maken van oplossingen voor vergelijkbare problemen die in het verleden gevonden zijn. Dit hergebruik is succesvol omdat gelijkende problemen meestal gelijkende oplossingen hebben.

Een CBR systeem bewaart de in het verleden opgeloste problemen (de casussen of "cases") met oplossingen in een database, de "case-base". Wanneer er een nieuw probleem opgelost moet worden, wordt in de case-base gezocht naar een oud probleem wat lijkt op het nieuwe probleem. Vervolgens wordt een oplossing voor het nieuwe probleem bepaald welke gebaseerd is op de oplossing van het oude probleem. Omdat het gevonden probleem meestal niet precies hetzelfde is als het nieuwe probleem is er een aanpassing nodig zodat het wel bij het nieuwe probleem past. Indien de nieuwe oplossing goed blijkt te zijn, kan het worden bewaard in de case-base om in de toekomst opnieuw gebruikt te kunnen worden. Door nieuwe cases te bewaren verbeterd een CBR systeem zijn prestaties — het leert.

Het in staat zijn om te leren is één van de vele voordelen van CBR systemen. Andere voordelen van CBR systemen in vergelijking met andere AI systemen zijn: een relatief eenvoudige kennisacquisitie (deze bestaat hoofdzakelijk uit het verzamelen van cases), een hoge betrouwbaarheid door in staat te zijn om nieuwe situaties te herkennen, en minder behoefte aan onderhoud omdat het systeem zich aan veel veranderingen in het probleemdomein zelf kan aanpassen. Als onderhoud al noodzakelijk is kan dit vaak beperkt blijven tot het onderhoud van de case-base, wat een tamelijk eenvoudige procedure kan zijn.

De voordelen van CBR maken het zeer geschikt voor NDO problemen, waar de kennis over data interpretatie vaak moeilijk te beschrijven is, betrouwbaarheid absoluut cruciaal is, en waar eenvoudig onderhoud om economische redenen belangrijk is. Omdat CBR systemen kunnen leren, kunnen ze zich aanpassen aan veranderende inspectie omstandigheden. Bovendien lijkt de normale NDO praktijk van data interpretatie zeer op de werkwijze van CBR systemen.

Binnen het ANDES project zijn twee prototype CBR applicaties ontwikkeld, één voor elk van de bovengenoemde test casussen. Resultaten van tests uitgevoerd op data uit echte veldmetingen waren tevredenstellend voor beide systemen. Het CBR systeem voor het URS wordt momenteel voorbereid om op de inspectie trein gebruikt te worden.

Tijdens de ontwikkeling van beide systemen vergden de kwesties van optimale case representatie en case vergelijking veel aandacht. Deze stappen zijn cruciaal voor het terugvinden van de meest gelijkende cases van de al geclassificeerde data in de case-base. Een verwant probleem is het vinden van de meest geschikte data voorbewerking ("preprocessing"), welke de data eenvoudiger maakt om te interpreteren, bijvoorbeeld door ruis en andere onwenselijke invloeden uit de data te verwijderen. Het verkrijgen van de data nodig om het systeem te testen bleek verre van triviaal te zijn. Verder eiste het URS systeem een hoge verwerkingssnelheid, zodat er een behoorlijke inspanning nodig was om tot een optimaal ontwerp van de case-base te komen.

Samen met andere systemen beschreven in de NDO literatuur bewijzen de twee ontwikkelde prototypes de geschiktheid van AI voor de interpretatie van NDO data. Dergelijke systemen worden echter zeer zelden voor daadwerkelijke inspecties gebruikt. Een van de redenen hiervoor kan de complexiteit van de factoren zijn welke een rol spelen in het NDO inspectie beslisproces. In principe is het mogelijk om de voordelen van het gebruik van (volledig) geautomatiseerde data interpretatie systemen in duidelijke economische termen aan te tonen. Zulke analyses zijn echter zeer ingewikkeld en hebben alleen zin voor specifieke gevallen; algemeen geldige analyses zijn niet uitvoerbaar. Een groot probleem bij de introductie van nieuwe technieken (niet alleen voor data interpretatie) in NDO is het feit dat bij een inspectie normaal gesproken ten minste drie partijen betrokken zijn: de producent van het inspectie systeem, het inspectie bedrijf dat de inspectie uitvoert, en de eigenaar van het geïnspecteerde product of de installatie. Succesvolle introductie van

een nieuw systeem vereist samenwerking van alle drie partijen en dit is niet eenvoudig te verwezenlijken. Wanneer geautomatiseerde interpretatie systemen goedkoper worden, eenvoudiger te ontwikkelen en gemakkelijker te gebruiken, zonder nadelen voor de betrouwbaarheid, zal hun introductie mogelijk minder weerstand ondervinden. Case-based reasoning is een techniek die dit kan helpen te bereiken.

Contents

Preface i
Summary iii
Samenvatting v
Contents ix

Chapter 1 Introduction 1
1.1 Nondestructive testing 1
1.2 ANDES project and its goal 2
1.3 Thesis overview 2
1.4 References 3

Chapter 2 Nondestructive testing 5
2.1 NDT inspection chain 5
2.2 NDT techniques: physics, applications, data, and interpretation 6
2.2.1 Eddy currents 8
2.2.2 Ultrasound 12
2.2.3 Acoustic methods 16
2.2.4 Radiation techniques 18
2.2.5 Thermal techniques 19
2.3 Quality of NDT 20
2.3.1 Inspection performance 20
2.3.2 POD and PFI 21
2.3.3 Cost aspects 23
2.4 Automating NDT inspection 26
2.4.1 Automated scanning 26
2.4.2 Automating NDT data interpretation 27
2.4.3 Typical existing automated inspection systems 29
2.5 References 29

Chapter 3 Artificial Intelligence 33
3.1 Understanding intelligence 33
3.1.1 What is intelligence? 33
3.1.2 Artificial Intelligence 34
3.1.3 Historical AI — the two paradigms 35
3.1.4 What *are* AI? 36
3.2 Expert systems 37

3.3 Artificial neural networks 39
3.3.1 Supervised trained networks 40
3.3.2 Unsupervised trained networks 43
3.4 Case-based reasoning 45
3.5 Other AI techniques 45
3.5.1 Reasoning with uncertainty 45
3.5.2 Machine learning 47
3.6 Choosing the right technique 49
3.6.1 Exploring a space of techniques 49
3.6.2 Maybe it does not have to be AI? 50
3.7 Building AI applications 51
3.7.1 Knowledge acquisition 51
3.7.2 Implementation 52
3.7.3 Maintenance 52
3.8 References 53

Chapter 4 Case-Based Reasoning 55
4.1 Introducing CBR 55
4.1.1 The origins 55
4.1.2 Main assumptions and ideas 56
4.2 CBR working cycles 57
4.3 Use of multiple types of knowledge in CBR 60
4.4 Advantages of CBR 62
4.5 Disadvantages of CBR 64
4.6 CBR as a lazy problem-solving method 65
4.7 Cases 66
4.7.1 Case representation 66
4.7.2 Indexing 68
4.7.3 Matching — the similarity measure 69
4.8 Case-base: organisation & case retrieval 70
4.8.1 Organisation 70
4.8.2 Retrieval 72
4.9 Adaptation 73
4.9.1 Adaptation done by the system. 74
4.9.2 Adaptation done by the user 74
4.10 Solution evaluation & case storage 75
4.10.1 Evaluation before use 75
4.10.2 Evaluation after use (feedback) 75
4.10.3 Case storage 76
4.11 Developing CBR systems 76
4.11.1 When CBR should be used 76
4.11.2 Design decisions 77
4.11.3 Knowledge engineering 79
4.11.4 System speed 80
4.11.5 Maintenance 80

4.11.6 Analysing CBR systems 82
4.12 CBR in practice 83
4.12.1 Application domains 83
4.12.2 Commercial applications 84
4.12.3 Examples of existing CBR tools 85
4.12.4 Future of CBR 86
4.13 References 87

Chapter 5 AI for NDT data interpretation 91
5.1 Possible uses of AI in NDT 91
5.2 Data interpretation 92
5.2.1 Distinguishing aspects of NDT data interpretation 92
5.2.2 Methods of data interpretation 93
5.3 Pattern recognition 94
5.3.1 Preprocessing 94
5.3.2 Feature extraction 94
5.3.3 Feature extraction in images 95
5.3.4 Classification 97
5.4 Use of various AI techniques in NDT data interpretation 102
5.4.1 Statistical and ANN classifiers 102
5.4.2 Expert systems 103
5.4.3 Reasoning with uncertainty 104
5.4.4 CBR 105
5.4.5 Hybrid systems 106
5.5 Developing AI systems for NDT 107
5.5.1 Characteristics of NDT data 107
5.5.2 Types of knowledge about data interpretation 108
5.5.3 Choice of AI technique 108
5.5.4 Factors deciding about success 110
5.5.5 AI and NDT standards 113
5.6 References 114

Chapter 6 Application 1: EC heat-exchanger inspection.. 119
6.1 Problem description 119
6.1.1 Inspection of heat exchangers 119
6.1.2 Eddy-current inspection 120
6.1.3 Eddy-current data from heat exchangers — typical signal responses 123
6.1.4 Data interpretation: problems, reasons for automation 130
6.2 Choice of AI technique 131
6.3 LISSA — introduction 132
6.3.1 Design specifications 132
6.3.2 Outline of the system 132
6.4 LISSA — general EC software functionality 133
6.4.1 Measurement data and system channels 133
6.4.2 Preprocessing 133
6.4.3 Detection 135

6.4.4 Frequency mixing 137
6.4.5 Phase calculation 138
6.4.6 Calibration 138
6.5 LISSA — CBR part 138
6.5.1 Case — short description 138
6.5.2 Case-base organisation and retrieval 138
6.5.3 Case matching 139
6.5.4 Case — detailed description 141
6.5.5 Threshold estimation 142
6.5.6 User interface 142
6.6 Tests 142
6.6.1 Test data and experiments 142
6.6.2 cal* files 145
6.6.3 tk* files 147
6.6.4 va* files 149
6.6.5 b* files 150
6.6.6 Discussion of the test results 151
6.7 Assessment 154
6.7.1 Fulfilling the requirements 154
6.7.2 Different way of working using LISSA 155
6.7.3 Deployment perspectives 156
6.8 Conclusions 156
6.9 References 157

Chapter 7 Application 2: Ultrasonic Rail-Inspection System 159
7.1 Problem description 159
7.1.1 Rail inspection 159
7.1.2 Defects in the rails 159
7.1.3 Ultrasonic Rail-Inspection System 160
7.1.4 Results of the old system 166
7.1.5 Requirements for the new system 166
7.2 Problem definition 166
7.3 Choice of AI technique — the argumentation 167
7.3.1 Rule-based classification 168
7.3.2 Choice of methodology for the new classification system 169
7.4 Data and image processing 170
7.4.1 Old image clustering 171
7.4.2 New image clustering 171
7.5 Case-based reasoning 172
7.5.1 Overall system design 172
7.5.2 Implementation 176
7.5.3 Case description 176
7.5.4 Case matching 177
7.5.5 Case-base organisation 179
7.5.6 Case retrieval 182
7.5.7 Match evaluation 184
7.5.8 Interaction with the user 185

7.5.9 Case-base maintenance 186
7.6 CBR system evaluation: experiments and results 186
7.6.1 Test data set 186
7.6.2 Test environment and software 187
7.6.3 Rule-based versus case-based versus hybrid classifier 187
7.6.4 Comparing various ways of case-base organisation 188
7.6.5 Results for the hybrid system 188
7.6.6 Disagreements in classification 192
7.6.7 Discussion of results 192
7.7 Conclusions 193
7.8 Future work 193
7.8.1 Deployment on the train 193
7.8.2 System improvements 194
7.9 References 195

Chapter 8 Discussion and Conclusions 197
8.1 Why and how to automate NDT data interpretation? 197
8.2 What is so special about interpretation of NDT data? 197
8.3 Which technique is best? 198
8.3.1 The requirements 198
8.3.2 Cost of construction 198
8.3.3 Cost of use 199
8.4 How does CBR fare? 199
8.4.1 Advantages and disadvantages 199
8.4.2 Issues when developing CBR systems 200
8.5 Problems in automating NDT data interpretation 201
8.6 Conclusions 201
Appendix A: Computer hardware prices 203
Appendix B: Publications 207
Subject index 209
Author index 215
Curriculum Vitae 219

Chapter 1

Introduction

1.1 Nondestructive testing

Inspection is an important aspect of the life cycle of industrial products. The scope of inspection ranges from examining simple articles like nuts and bolts to checking complex constructions like nuclear power plants and railway networks. In general, inspection is carried out to see whether a product or its parts fulfil certain requirements. These requirements usually reflect the compromise between the desired product properties and the price one is prepared to pay for it.

Inspection begins during the product development phase, especially where mass-produced products are concerned. Tests are done to determine the relationship between the parameters of the manufacturing process and the resulting product quality. During manufacture inspection can be carried out either continuously or by means of random sampling.

With some products, inspection continues after manufacturing has finished. One reason for having in-use inspection is that it may extend the lifetime of the product. In such cases, the inspecting is usually done to establish whether maintenance is necessary and, if so, what kind of maintenance needs to be done. Of course, typically periodic maintenance does not need to be preceded by inspection. Having a preference for one system or the other will be determined mainly by the cost aspects.

Another reason for in-use inspection is to determine when a product is reaching an end of its lifetime. This is important in situations where a sudden breakdown may incur significant costs. An extreme example of such situation is an aeroplane crash caused by structural failure.

Finally, a product can be examined after it has been taken out of use. This can be done, for example, to determine the reason for failure (if such occurred) and to provide feedback for product improvement.

One way to differentiate between the various inspection types is by distinguishing between destructive and nondestructive testing (NDT). In the case of destructive inspection the product gets damaged during the actual test procedure, an example being cutting through a vessel to see the inside. Nondestructive testing does not affect the inspected object. (There is also an intermediate type, for example, when for the purpose of inspection a construction has to undergo expensive dismantling or when samples are taken from a product.)

Most product testing is does not affect the product, however, such testing is rarely explicitly called nondestructive. The term *nondestructive* is explicitly used to indicate an inspection type where for a good insight into a problem a destructive inspection method would ideally be required. However, using a special inspection technique almost the same information about the product can be obtained without damaging it. Moreover, the term NDT is usually used for after-production inspection in contrast to in-production process monitoring.

Nondestructive testing can be carried out using various techniques, like: X-rays, ultrasound, eddy currents, and acoustic emissions. The choice of an inspection technique depends not only on its capabilities, for instance, for defect detection, but also on the compromise achieved between the quality of the data delivered and the cost of inspection.

Often, the data from nondestructive inspection does not directly show the presence of defects nor does it reflect the quantity one wants to measure but, instead, requires some processing and interpretation to determine if the inspected object is flawed or not. The data interpretation is usually done by an experienced NDT inspector.

Increasing automation in the inspection hardware has made it possible for more inspecting to be done but has also increased the amount of data to be analysed. Using more sophisticated inspection techniques makes it possible to improve defect detection ratios but it often also increases the complexity of the data that has to be analysed. Both factors make it necessary to look for automated data analysis methods. Increases in processing power together with decreases in the size and price of computer systems, and the emergence of digital-storage inspection systems, make such automatic data interpretation even more feasible.

Apart from increasing the speed of inspection, using automatic interpretation techniques increases the reliability and consistency of inspection. In addition, automatic systems can help the operator to interpret complex data by, for instance, doing advanced data preprocessing or analysing multiple data streams simultaneously.

1.2 ANDES project and its goal

The ANDES (Automatic NonDestructive Evaluation System) project has been started at TNO Institute of Applied Physics after encouraging results from a research project into the use of neural networks for defect type classification based on the ultrasonic images of welds [Lorenz, 1993; Wielinga et al., 1994]. The intention was to further investigate automatic NDT data interpretation methods. The work has been done in cooperation with Delft University of Technology and it has been sponsored by Shell, Akzo-Nobel, NS Ultrasoonbedrijf and Materiaal Metingen Testgroep BV. The sponsors have provided two test cases for which automatic interpretation software had to be developed: eddy-current inspection of heat exchangers and ultrasonic rail inspection. Both cases show characteristics typical to other practical inspection problems. The data is inhomogeneous, i.e., it may originate from combinations of defects, construction elements, and noise sources. A-priori information on the inspected constructions and the expected defect indications is limited. Furthermore, fluctuations in inspection conditions have to be handled, as well as differences between calibration results and real defect indications.

The complexity of the problems, the fact that the interpretation is successfully performed by human inspectors, and the research already done into this problem, suggested that Artificial Intelligence (AI) could offer a set of techniques and methodologies best suited to solving the problem. Thus the objective of the project was *to determine which AI technique or which combination of techniques would give the best results when applied to common NDT problems.* The "best" technique would be quantified by factors like: the cost of development, manufacture and maintenance, the recognition ratio, reliability, general applicability of the technique used, etc. The chosen approach would be verified by developing and testing automatic interpretation software for the two test cases. While developing the interpretation systems experience would be gained, which would hopefully provide more insight into the general problem of developing automatic interpretation systems for NDT data.

1.3 Thesis overview

This study involved two disciplines, therefore, the thesis begins with two introductory chapters. Chapter 2 gives an introduction to the field of nondestructive testing. Various inspection techniques are presented giving some physics background, the types of inspection problems for which they can be used, and the types of data they deliver. Next, the methods of describing and evaluating the performance of NDT methods and systems are presented. This includes the issue of the cost of doing (or not doing) NDT. Some preliminary considerations about automating NDT inspections are also given.

In Chapter 3 an introduction to the field of Artificial Intelligence is given. The chapter begins with a general definition of AI. Next, several AI techniques are presented: the expert systems, artificial neural networks (ANN), and case-based reasoning (CBR). Also, means of dealing with uncertainty (for example, fuzzy sets and fuzzy logic) as well as Machine Learning (ML) techniques are

handled. Because in the existing automatic NDT systems progress can be seen from the use of statistical classifiers to the use of various types of neural networks, Chapter 3 also briefly discusses relation of statistical classifiers and ANNs. Next, the problem of choosing the right technique for a given problem is discussed. Finally, the problem of developing and maintaining AI applications is treated.

Most research within the project concentrated on case-based reasoning (CBR). Chapter 4 present this technique in more detail. First, the conceptual framework behind CBR is presented. Next, the variations of CBR systems and the various ways of describing their working are presented. Then, more details concerning the cases, their representation, matching, storage, retrieval and adaptation are described. After giving the theoretical background, the development of CBR systems is described. Several representative existing CBR tools are mentioned, as well as some applications of CBR.

Chapter 5 discusses the use of AI techniques for NDT data interpretation. It begins with a presentation of general classification and pattern recognition techniques. Consideration is given to preprocessing, feature extraction, clustering, etc. Next, various AI techniques for NDT data interpretation are discussed and various examples of academic and commercial systems are presented. The chapter ends with considerations about development of AI applications for NDT, including influence of AI on NDT standards and regulations.

Chapters 6 and 7 describe the results of research done on the two test cases: eddy-current inspection and the ultrasonic rail-inspection system. In both chapters first the inspection problem is described. Then, the chosen methodology for the development of the automatic interpretation system is given. Next, the developed prototype systems are described and the results of the tests are presented. Both chapters end with a discussion of the results and possibilities of deployment of the software.

Based on the study of the existing systems and on the experience with the development of the software for the two test cases Chapter 8 discusses the use of AI techniques for the development of automatic NDT data interpretation software. Special attention is paid to the use of CBR as this technique has so far been seldom used for NDT problems. Attention is paid not only to the development of prototype systems but also to the problems encountered in the deployment of these systems in the field.

1.4 References

Lorenz, M. (1993) *Ultrasonic Imaging for the Characterization of Defects in Steel Components*, Ph.D. thesis, Delft University of Technology, The Netherlands.

Wielinga, T.S., Kerckhoffs, E.J.H., and Lorenz, M. (1994) "Towards the classification of round and planar weld defects with neural networks", *Neural Network World*, Vol. 4, pp. 465–478.

Chapter 2

Nondestructive testing

The purpose of testing is to determine if a product or installation (still) fulfils its specifications. One way to distinguish testing techniques is into destructive / nondestructive categories. Nondestructive testing will leave the tested object unaffected while destructive testing might leave it unsuitable for further use. There is a "gray area" in this distinction, for example, where samples are taken from a product to carry out a chemical analysis or where an object is not destroyed but the inspection may have required an expensive disassembly.

Not all measurements on a product or installation are denoted as NDT measurements. The term nondestructive testing will usually be used in situations where special techniques are required to do internal inspection. A simple measurement of outside dimensions, for example, though being nondestructive will not be explicitly called as such. Also visual inspection is generally not considered an NDT technique, although it can be very effective.

This chapter describes the most common nondestructive testing techniques, presents some methods of evaluating their performance both as far as the results and as far as the inspection cost are concerned, and presents ways of automating NDT data interpretation.

2.1 NDT inspection chain

A typical NDT setup (processing chain) is shown in Figure 2.1. Inspection is performed on a specimen to which some form of energy (for instance, ultrasonic pulses or X-rays) is applied and the interaction of that energy with the specimen is picked-up using a sensor. The signal from the sensor is measured using an acquisition system. The acquisition system may also record location (positioning) information together with the measured signal. Afterwards, the signal may be processed and presented in a form most suitable for the interpretation.

As indicated in Figure 2.1, the interpretation stage of the NDT chain typically consists of detection, classification, and characterisation:

- The *detection* involves selecting those parts of the data stream which may contain indications from defects. Defined this way it is not equivalent to defect detection — for example, the detected signal may be due to construction elements. The detection is usually triggered by dif-

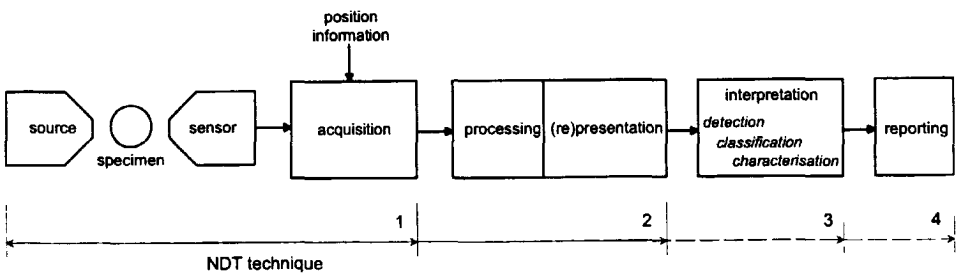


Figure 2.1: NDT inspection chain. By NDT technique one usually means the combination of the energy source, the sensor and the acquisition technique used. For many modern techniques, for example X-ray tomography, the data coming from the acquisition system requires complex processing before it is in a form of an interpretable representation. In the figure, four levels of integration are depicted — full size systems will cover all aspects, whereas for simple measurements only acquisition hardware is necessary.

ferences between the current signal and the signal from a uniform, non-defect part of the specimen. (It is assumed that inhomogeneities in the data stream will correspond to inhomogeneities in the specimen, and these, in turn, can be caused by defects.) The detection is not always explicitly present, for instance, if discrete measurements are made then all of them may be subject to classification and characterisation.

- The *classification* means assigning the data to one of a number of predefined categories, for example, distinction between defect, baffle, and tube sheet in eddy-current heat-exchanger inspection; or distinction between planar and spherical defects in ultrasonic inspection of welds. No classification is needed if all detected indications can be assumed to come from the same type of defect.
- The *characterisation* means determining relevant parameters of the detected feature. In simple situations characterisation may be done without prior classification; however, often, knowledge of the class (type) is necessary for correct characterisation, for instance, in situations when a number of calibration curves (see further) are used, each applicable to a certain defect type. Usually, characterisation implies defect sizing, which, together with prior classification, provides information about the severity of a defect.

Results of the interpretation have to be reported. Often, only the most serious defects are reported as these determine whether the inspected object will be repaired or replaced. Modern inspection systems make reporting easier by directly interfacing with reporting software, which can present the inspection results using spreadsheets, graphics, etc.

More and more inspection is done using digital equipment which allows for storage of data at the various stages of the inspection chain — usually, after some initial preprocessing. This makes it possible to break the inspection chain and, for example, to do the data acquisition at an inspection site and do the processing and interpretation at a lab. If all the data is recorded it is possible to compare consecutive inspections and to analyse trends.

Calibration. In some NDT techniques the defect characterisation can be done directly based on the obtained data. For example, in the majority of applications of X-rays the measurements of the size of the defects can be done directly in the image obtained on film. For other techniques it is possible to construct a more or less accurate model for the inspection so that the obtained signal can be related to the characteristics of the defect and vice-versa (though the inversion is not always unequivocal, as for instance in the case of eddy-current inspection). However, usually, signal characterisation is done based on the comparison with signals obtained from the inspection of specially prepared calibration pieces with known type and size of the defects. The signal obtained from the real inspection can be related to the signal obtained during the calibration and the defects can thus be characterised.

2.2 NDT techniques: physics, applications, data, and interpretation

Nondestructive testing can be done using a variety of methods, which can be grouped in various categories. Table 2.1 shows one of the possible (not exhaustive) categorisations distinguishing optical, electromagnetic-electronic, sonic-ultrasonic, penetrating radiation, and thermal techniques.

In the next few sections of this chapter some of the techniques from Table 2.1 are described in more detail, namely: eddy currents, ultrasound, acoustic methods, radiation techniques, and thermal techniques. These techniques have been chosen because they are widely used and they demonstrate a wide variety in data types and data analysis problems. Description of each technique begins with the presentation of the physical basis of the technique. Then, the application area is described, followed by description of the typical data obtained using the technique and brief remarks about the ways of data interpretation.

Table 2.1: NDT categories (a selection based on [Wenk et al., 1987]).

Technique	Physical principles	Use	Data & interpretation
optical			
visual-optical	visual (direct or optically aided — e.g., endoscopes) inspection of object surfaces in visible or ultraviolet light (for fluorescent materials)	detection of cracks, pores, estimation of roughness	C-scan, image interpretation, measurement in image
liquid penetrant	a penetrating liquid enters surface-connected cracks, after removal of excess liquid a developer is applied which causes the penetrant to bleed out	cracks, leaks	image, visual observation
electromagnetic-electronic			
magnetic particle	magnetic powder is applied to surface of a magnetised object, powder accumulates or creates patterns	crack, pores, inclusions, permeability variations	visual inspection, image analysis
magnetic flux leakage	magnetic field is applied to the object, the leakage of the field caused by subsurface inhomogeneities is detected using magnetic-field probes	crack, pores, inclusions, permeability variations	(complex) signal, C-scan
eddy current	localised alternating current loop (eddy) is induced in test object, change in inductive reactance of the probe to magnetic field of induced current indicates flaws	cracks, pits, inclusions, thickness (wall, coating), conductivity, permeability	scalar or complex value, complex signal (oscilloscope image), C-scan
electric current	current flows through part under test, current strength between electrodes touching surface is affected by inhomogeneities	cracks and inclusions, material structure variations, resistivity, corrosion, fatigue damage	scalar value
microwave radiation	continuous or modulated radiowave radiation directed at object propagates according to internal state or structure of part	cracks, holes, debonds, inhomogeneity, thickness, dielectric properties, moisture content	scalar value
sonic-ultrasonic			
acoustic impact	striking, tapping or hitting an object causes vibrations which can indicate anomalies, inspection can be local or global	cracks, debonds, delaminations, variations in physical dimensions, mechanical properties	scalar value (signal strength), audible sound (spectrum), acoustic wave signature
acoustic emission	energy released at deformation or fracture sites causes (ultra)sonic waves to propagate in material	crack initiation and propagation	event counter, event coordinate plot
acoustic (vibration) monitoring	sound emanating from working mechanical parts	wear, misalignment	frequency spectrum, waveform signature
ultrasonics	ultrasonic pulses are directed into test object, ultrasonic echoes indicate absence, presence and location of flaws, and discontinuities	cracks, laminations, debonds, inclusions, porosity, grain size, thickness, density	A-scan, gating, B-scan, C-scan
penetrating radiation			
X-ray radiography	X-ray radiation transmitted by test object is used to image internal structure and/or flaws	cracks, voids, porosity, inclusions, malstructure, misassembly, density variations	image analysis
gamma radiography	radiation emitted by isotope source transmitted by test object is used to image internal structure and/or flaws	cracks, voids, porosity, inclusions, malstructure, misassembly, density variations (also in thick materials)	image analysis
thermal			
thermal imaging	a static thermal image or the rate of heating or cooling of the object is observed using infrared camera	delaminations, debonds, thickness,	image (sequence) analysis

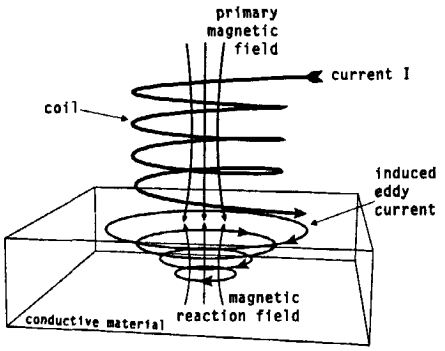


Figure 2.2: Alternating-current coil inducing eddy currents in a specimen.

2.2.1 Eddy currents

Theory. The eddy-current (EC) testing method relies on the principle of magnetic induction. When a coil excited by an alternating current is brought in proximity with a conductive material it induces eddy currents in that material. The direction of the induced eddy currents, and of the reaction field generated by these currents, is such as to oppose the change in the primary field (see Figure 2.2). If the specimen is non-ferromagnetic then the inductance L of the coil decreases (compared to its inductance in the air). At the same time, the resistance R of the coil increases. If the specimen is ferromagnetic then both the inductance and the resistance increase.

Presence of a discontinuity or inhomogeneity in the specimen causes a change in the induced eddy currents. As a consequence, the inductance and the resistance of the coil also change. These variations in the coil impedance can be measured and used for various purposes: detection of defects, material characterisation, distance measurement, etc.

Eddy currents have a limited penetration depth. The amplitude of the eddy currents decreases exponentially beneath the metal surface. The depth of penetration decreases with the frequency f. This so-called skin effect limits the use of EC testing techniques to relatively thin materials.[1] Apart from the amplitude of the eddy currents, also their phase changes with depth. In thick samples the change is linearly proportional to the depth. Both amplitude and phase of the currents obstructed by a flaw influence the amplitude and phase of the test coil impedance. This makes it possible to distinguish between, for example, small surface defects and large internal defects. These could have the same effect on the magnitude of the test coil impedance, however, the phase of the impedance will be different.

Applications of EC. The typical uses of EC technique are for detection of defects in heat-exchanger pipes, inspections of riveted connections in aeroplanes, detection of cracks in turbine blades and aeroplane wheels, etc. Eddy currents can also be used for measuring mechanical and metallurgical properties which correlate with electrical and magnetic properties. Advantages and disadvantages of the EC technique are listed in Table 2.2.

1. Cecco et al. [1983] give the following formula for the skin depth d in mm at which the eddy-current density has attenuated to $1/e$ (37%) of the surface density:

$$\delta = 50\sqrt{\frac{\rho}{f\mu_r}}$$

where ρ is the electrical resistivity in mW/cm, f is the test frequency in Hz, and μ_r is the relative magnetic permeability (dimensionless).

Table 2.2: Advantages and disadvantages of the eddy-current technique.

Advantages	Disadvantages
• sensitivity to small defects, • fast inspection (large inspected area of the probe, fast scanning speed), • simple equipment necessary to do inspection, • no need to prepare inspected surface.	• can inspect surface area only, due to the skin effect, • sensitivity to a large number of material properties which may influence inspection results, • difficult analysis of composite indications, • no direct inversion of results.

Data acquisition systems. The measured impedance Z is a complex quantity:

$$Z = R + j\omega L$$

with resistance R and inductive reactance ωL. Resistance and inductive reactance can be plotted in the complex plane, see Figure 2.3. In the complex plane the impedance can be characterised by its phase and amplitude. The earliest EC testing equipment could display the amplitude of the impedance, but could not detect impedance-phase changes. End 1970s instruments became available which could display both amplitude and phase, i.e., both the inductive reactance ωL and resistance R components separately [Hagemaier, 1983]. The impedance plane curves were drawn on an integral X-Y storage oscilloscope. End 1980s instruments were introduced which used computers to display the data and to do simple processing tasks.

It is possible to measure either impedance of a single-coil probe or a difference of the impedance between two probes placed close to each other. Choice for either the absolute or the differential (or both) measurement is determined by what types of defects are expected — large or localised (see Section 6.1.2).

The main controls on eddy-current test equipment are for adjusting the test frequency, phase, and gain, and for balancing (zeroing the signal). The test frequency is usually chosen such as to provide best phase separation between inside and outside defects. The phase adjustment can be used to rotate the signal so that it is easier to interpret, for instance, indications caused by probe wobble are usually aligned along one of the axes.

Digital inspection systems digitise the signal before they process it. The signal sampling can be done at equal time intervals or it can be triggered by a position encoder. Position-based sampling usually requires use of a scanner or a push-puller (a motor driven device which pulls or pushes a cable with the probe through a tube) which may sometimes be inconvenient to use (for example, in cramped spaces); however, position sampling allows for easier and more advanced analysis of the data. In case of time sampling the probe is moved by hand, which may be faster and more convenient than a use of push-puller; however, some data can be ambiguous or difficult to interpret, especially when the probe speed was not kept constant.

Usually, the results of an inspection are analysed on-the-fly and no data is stored. Before the use of digital equipment has become widespread, when necessary, the inspection used to be recorded on strip charts. Digital equipment makes it easy to save the inspection data. However, because of the size of the data files often only a selection of the data is stored.

Signal processing. A single frequency signal from an eddy-current probe may be difficult to interpret if it comes from a combination of a defect with, for example, a baffle or a dent in a tube. If, however, two (or more) signals with different frequencies are measured then it is possible, by mixing of the signals, to remove most of the unwanted baffle or dent signal leaving only the signal from the defect. This method is only effective if the defect signal differs from the unwanted signal and if signals are vectorially additive. The first condition means that detection of an internal defect in the presence of internal variations is impossible. The second condition means that detection of defects under non-ferromagnetic baffle plates is impossible.

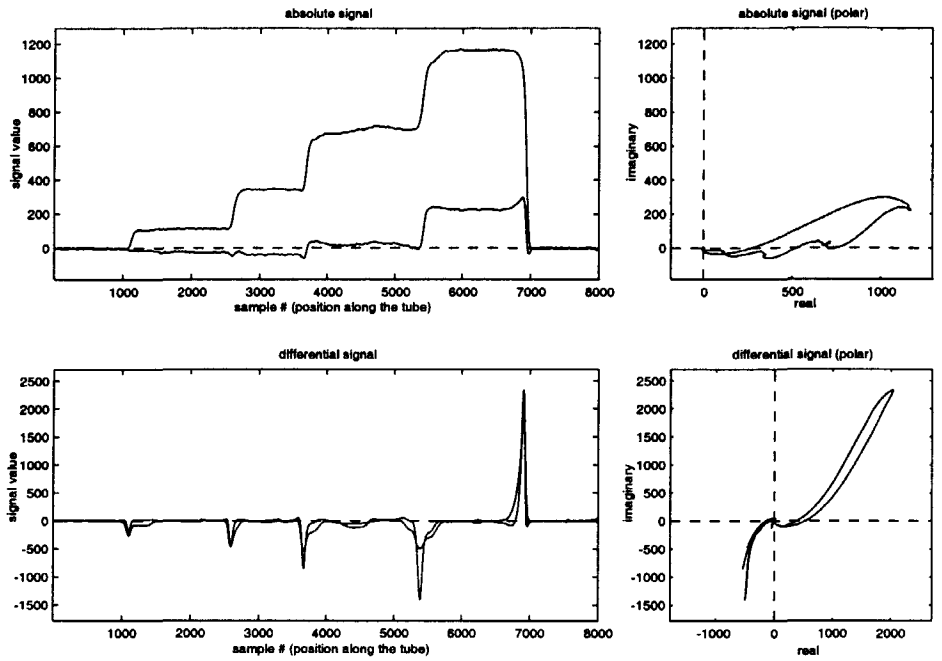


Figure 2.3: Plots of EC data: top – absolute signal, bottom – differential signal, left – line plots (real and imaginary impedance components), right – impedance (polar) plane plots (showing Lissajous curves). Usually, values of the measured signal are interpreted in relation to the values of the calibration signal, moreover, the signal is usually phase-rotated, that is why no particular meaning is assigned to the values on the real and imaginary axis.

Mixing is done by shifting, scaling, and rotating one frequency signal and subtracting it from another frequency signal. The shift, scale and rotation parameters are chosen such that the result of the mix of the unwanted signal (the residual) is as small as possible. In older equipment the mixing parameters (gain, phase, and offset) had to be set by hand; the newer computer-based systems can calculate the mix parameters automatically from a baffle signal, for instance, using least-squares estimation [Stolte et al., 1988] or numerical minimisation (as used in LISSA described in Chapter 6).

EC data presentation. Data from eddy-current measurements is usually presented as line or polar plots (see Figure 2.3). A line plot shows the real and imaginary component of the impedance signal as a function of time or probe position. A polar plot displays the impedance signal in the complex plane. The curves formed by the signal in the polar plot are called Lissajous curves — this type of signal would be seen on the old equipment using analog oscilloscopes.

Line plots are useful for locating the defects. The absolute line plot can be used for the detection of long defects. The differential polar plot shows the most complicated signal, whose form and orientation can be used to distinguish between various types of defects. When a surface scan (C-scan) is made with an EC probe then the signal may be represented as signal traces or as a colour-coded plot, see Figure 2.4 and [Smith, 1998].

Data interpretation. Typically, as in many NDT techniques, data interpretation is based on calibration measurements. The simplest way to interpret EC signals is to look at the signal amplitude. This method works well if the damage mechanism for all defects is the same, for example, widespread outside corrosion. If the damage can occur on both outside and inside of the specimen, then

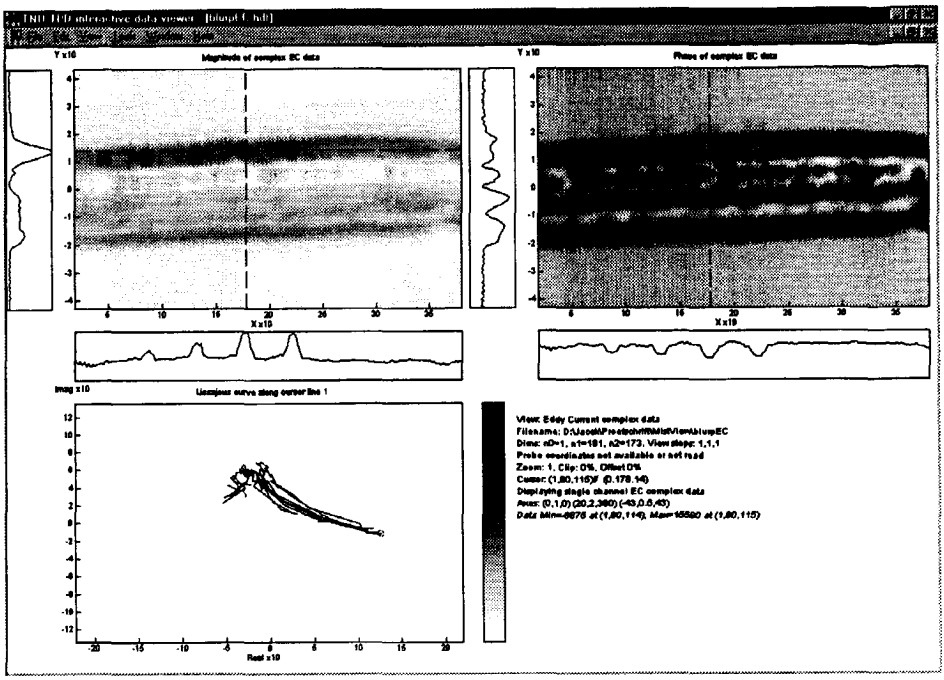


Figure 2.4: Presentation of eddy-current data from a surface scan. Top-left shows amplitude of the complex impedance signal, top-right shows the phase. The narrow plots at the edges show the values along the x,y lines marked in the surface views. Bottom-left shows values of the signal along the x line (fixed y) presented in the complex impedance plane, this is the Lissajous curve.

also the phase of the signal has to be taken into account in defect characterisation. More precise characterisation can be obtained if also the shape of the signal is analysed, if necessary, at several frequencies simultaneously. Though there are models of the EC phenomena available, their use for data characterisation is hampered by the large computational cost [Monebhurrun et al., 1995]. Moreover, because the probe (especially the bobbin probe) integrates the effects over a large (compared to the size of defects) area, exact characterisation of the defects is often not possible.

Related Techniques. There are several other electromagnetic techniques similar to the eddy current technique [Krzywosz & Dau, 1990]. Similarity lies, apart from the use of similar electromagnetic phenomena, in the inspection procedures, in the data obtained, and in data interpretation. These techniques can be used when the normal EC technique fails, for instance, to inspect ferromagnetic or thick specimens. They are:

- *Magnetic saturation EC technique* — uses DC electromagnets or permanent magnets to saturate ferromagnetic material (which cannot be inspected using the normal EC technique) so that it has a relative permeability of one. The material then behaves like a non-ferromagnetic material and permeability variations no longer affect the results. Once a saturating field has been applied, the normal EC technique can be used.
- *Remote field eddy-current (RFEC) technique* — uses a detector coil which is placed at a distance (two to three pipe diameters) away from the exciter coil. Changes in the phase of the signal picked up by the detector coil are used to determine the wall loss. RFEC can be used to inspect specimens thicker than EC technique could handle. The frequencies used are lower than in a normal EC inspection which limits inspection speed. Because the signal in the detector is low, noise

heavily influences accuracy of the results. Unlike the standard EC method, RFEC cannot distinguish between outside and inside defects, moreover, the spacial resolution is lower. RFEC is often used to inspect large, thick steel plates, like for example, bottoms of storage tanks.

- *Magnetic flux leakage (MFL) testing* — is done by magnetising the object to be tested and scanning the surface with a flux-sensitive detector. Discontinuities present in the tested object perturb the flux and thus are detected. MLF technique is less sensitive to lift-off than normal EC is, it is suitable for ferromagnetic materials, and it has better spacial resolution than RFEC. It is often used to inspect large-diameter pipelines.

2.2.2 Ultrasound

Theory. A common NDT technique is ultrasonic testing [Birks et al., 1991]. In ultrasonic testing an ultrasonic wave is effectuated in the inspected specimen by means of an ultrasonic transducer and the interaction of the wave with the interior of the specimen is monitored. An ultrasonic wave propagating through a solid object undergoes various interactions like mode conversion, reflections, diffraction, and attenuation. The mode of the wave defines the way the acoustic wave propagates, as characterised by the particle motion in the wave. One distinguishes shear, Lamb[1], surface, and longitudinal waves. Mode conversions occur at the interfaces of various materials depending on the type of material and the angle of incidence. Reflection occurs mainly when the wavefront hits structures larger than the wave length, in particular interfaces between materials. The diffraction occurs at the edges of structures, especially when the size of these structures is comparable with the wave length. Attenuation depends on the mechanical properties of the inspected material like the modulus of elasticity and the grain size.

These effects, either single or in combination, can be used to measure properties of the inspected object. Applications of ultrasonic testing include:

- discontinuity detection — defects (voids, cracks) in materials,
- thickness measurement — e.g., to determine the material loss due to corrosion,
- determination of elastic moduli — e.g., to monitor product quality,
- study of metallurgical structure, etc.

The setup. Ultrasonic testing is typically done in one of two ways:

- *Reflection technique* (usually in pulse-echo mode) — with transmitter and receiver on the same side of an object, see Figure 2.5. A high-frequency pulse is emitted into the inspected object and the returning signal (echoes) is received and displayed. Usually, the received signal is dis-

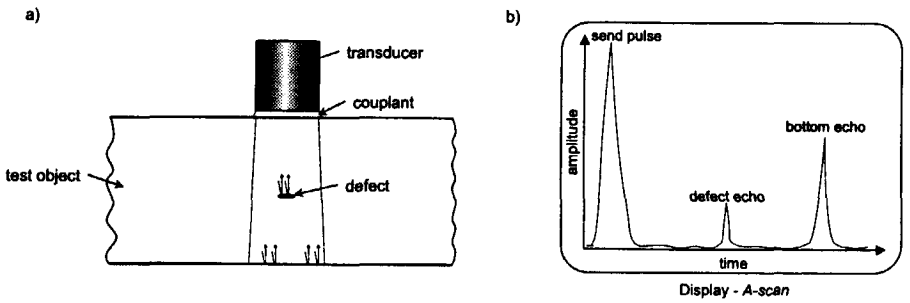


Figure 2.5: Principle of ultrasonic pulse-echo examination (a) and the resulting A-scan (b).

1. Lamb wave (plate wave): A type of ultrasonic wave propagation in which the wave is guided between two parallel surfaces of the test object (from *Ultrasonic Testing Encyclopedia* at www.ndt.net).

played as a function of time (so-called A-scan), but other representations can also be used (see further).

- *Transmission technique* — with a transmitter on one side of an object and a receiver on another. In this technique usually the wave attenuation is measured.

A typical ultrasonic setup consists of a transmitter, a couplant to transfer the acoustic energy into the object, and a receiver. A couplant is needed because transmission through interfaces of materials with large difference in wave speed (for example, air and steel) is very low. The transducers are usually piezoelectric. A combined transmitter-receiver is typically used because of the lower cost and ease of handling. A setup shown in Figure 2.5 would have problems with detecting cracks perpendicular to the surface, that is why often angled probes are used, the angle depending of the expected defect orientation. Advantages and disadvantages of the ultrasonic technique are listed in Table 2.3.

Table 2.3: Advantages and disadvantages of the ultrasonic technique

Advantages	Disadvantages
• high sensitivity — detection of small discontinuities, • good penetration — inspection of thick pieces, • accuracy in defect location and size determination, • relatively fast (indications obtained in real-time), • needs to access only one side of the object, • in many applications results are easy to interpret, e.g., measuring material thickness.	• difficult to use on objects with complex surface geometry, because of size of the probe, • need to use a couplant (may require immersion in water), • difficult to use on objects with grainy, porous, etc. internal structure, • possibility of missing defects located at certain angles to the beam, • difficulty in interpretation of signal with complex reflection and interference patterns, e.g., scans of certain types of welds, • difficulty in determining type of the defect, e.g., distinction between planar and spherical defects.

Data form. Data from the ultrasonic inspection is usually presented in one of the following forms (see Figure 2.6):

- *A-scan* — Typical for pulse-echo technique. An A-scan shows the amplitude of the signal as function of time, see Figure 2.5b. For a high signal-to-noise ratio (SNR) any reflections from the discontinuities will be clearly visible, making it possible to determine the depth of the defect. The amplitude of the signal can be a measure of the area of the reflector, if the discontinuity is smaller than the beam size.
- *B-scan* — While an A-scan gives a 1-dimensional view into the inspected object, a B-scan makes it possible to obtain a sort of 2-D cross-section of the inspected object, see Figure 2.6. A B-scan is obtained when a probe is moved along a line over the object and the resulting A-scans are displayed next to each other. The amplitude of A-scan signals is usually represented as intensity or colour values. The signal may also be gated showing only echoes above certain amplitude. B-scans can also be used to combine data acquired using several transducers, each located under a different angle. In that case, data is transformed to common reference coordinates. (This type of data presentation is used in the URS system described in Chapter 7.)
- *C-scan* — typically presents a surface view of the scanned object. Values (represented as shades of grey or colours) of the points in a C-scan represent values of the measurements at the corresponding points on the specimen. For pulse-echo measurements these will typically be the values of the maximum echo amplitude or the time to the first (or maximal) echo signal. For the transmission measurements these will be the values of wave attenuation.

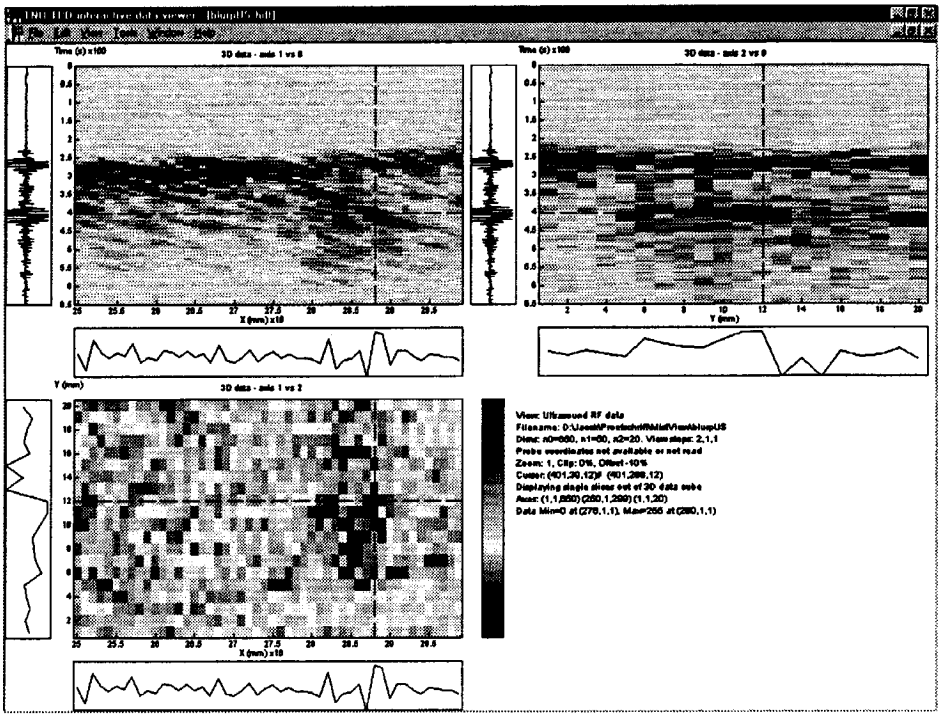


Figure 2.6: Presentation of ultrasonic data: top-left & top-right – B-scans in perpendicular directions (the right one is stretched), at bottom – a C-scan showing values of the data at a certain depth. The narrow plots show data values along the marked lines, the top vertical ones show a single A-scan signal.

Modern inspection equipment makes it possible to acquire the whole A-scan signal at each measurement point. If a surface scan is done and the A-scan values are combined then a 3-D data set is obtained which provides a 3-D view of the inspected object. However, 3-D visualisation only recently has become feasible and still may be too complex for field use. 3-D views may also be difficult to interpret. Typically, such data is analysed by looking at 2-D slices which correspond to B-scans and C-scans, see Figure 2.6. For 3-D data to be really useful it is required that the inspection be done with an automated scanning system to obtain accurate coordinate information.

Signal processing. Raw ultrasonic data may be sometimes difficult to interpret by the operators. This is especially the case for low SNR signal. Because of that, (pre)processing of ultrasonic signal is often done to facilitate signal interpretation. Examples of processing steps include:

- *Averaging* — done to improve the SNR. It is possible to can add several measurements made at the same spot. Assuming that the noise is random, the amplitude of the signal from the defects will increase above the noise signal. However, this technique cannot be used in fast scanning systems or when the same location of measurements to be averaged cannot be guaranteed.
- *Signal filtering* — done to remove unwanted signals, usually, noise. Most common method of filtering is to remove signal in certain frequency bands.
- *Deconvolution* — The detected signal is a convolution of the input signal with the impulse function of the medium [Zala et al., 1988]. The aim of deconvolution is to aid the interpretation of the signal by reducing signal complexity caused by a complex form of the input pulse. Decon-

volution helps to separate signals from closely spaced defects and simplifies analysis of phase changes in the echo signal (a phase change can indicate, for instance, the type of interface causing a reflection).

Data interpretation. The way the data is interpreted depends to a large degree on the form in which the data is presented. The typical interpretation methods for the most common forms of data presentation are given in the following.

A-scan

The simplest form of interpreting ultrasonic data in A-scan form is gating. A threshold area or line can be defined on the A-scan display, separating areas where signal from a non-defect piece is located from areas where signals from defects would be visible. When a signal enters the threshold area or crosses the threshold line then an alarm is triggered. This method can be used, for example, to detect defects or an abnormal wall-loss. However, in its simple form it will not work well if the SNR is low. Gating usually has to be supplemented by measurements in the signal being displayed (once an alarm has been triggered). One can measure signal amplitude (to estimate the defect size) or time (to estimate the distance to the reflector).

An A-scan echo signal will look differently depending on the type (form) of the defect that caused the echo. An experienced inspector may be able to determine the defect type based on the form of the signal. Use of automatic interpretation systems allows for more advanced signal analysis, looking at various data representation forms in order to determine indication type. [Adler et al., 1991] list a number of data representations which can be sources of features for automatic signal interpretation, for example (to give an idea of the variety of the possibilities): time domain profiles of a waveform; power, analytical spectrum, or phase in the frequency domain of the signal; transfer function domain using the initial pulse of the transducer as the reference; reflection factor profiles as a function of angle; dispersion curve profiles; etc.

B-scan

Interpretation of the signal may be easier if several A-scans are combined into a B-scan. Abnormalities which could come from defects, as well as indications of construction features like a changing profile, holes, etc. are then easier to recognise. Interpretation is easier because a single defect may be visible on several neighbouring A-scans. When the data comes from an automated scan, it may be possible to display the data on the outline of the inspected part, making it easier to recognise any artefacts coming from the construction features and also to concentrate the attention on the areas where occurrence of the defects is most probable.

Still, analysis of the B-scans may be quite complex due to superposition of various reflections in the image, diffraction effects, etc. In such situations interpretation is done based on the images from calibration pieces. The operator learns how the images from non-defect specimens, as well as from typical defects look like, and uses this experience to interpret the image. Thus, interpretation will be done mainly based on the abnormalities observed in the images.

Presentation of data in B-scan form usually requires digital storage of data. This opens a possibility of interactively adjusting the display parameters (use of various thresholds and colour maps) possibly allowing for spotting small defects within noise.

Knowing the geometry of the inspected piece and using automated scanners makes it possible to take the geometry into account when analysing the data. Ray tracing or other techniques can be used to support interpretation.

C-scan

The C-scan representation of data is especially useful for detection of abnormalities in normally regular patterns in the inspected data, for example, in composite reinforced materials. It is also convenient for determination of areas of widespread damage as, for example, caused by corrosion. Interpretation is usually helped by segmenting the image by a use of thresholds and appropriate colour

maps. If a C-scan represents a view into complex 3-D data interactive analysis may be required. In such a case, an operator makes a selection of the most appropriate aspect of the 3-D data which will be displayed as a C-scan.

Analysis of ultrasonic data may also be done in frequency domain. Variation in the structure of the inspected object will be visible as changes in the frequency spectrum.

2.2.3 Acoustic methods

As listed in Table 2.1 there are three variations of the acoustic methods: acoustic emission methods, impact methods, and acoustic (vibration) monitoring methods. All these techniques register sound emanating from the inspected object and are, to a certain degree, overlapping. The difference lies in the cause of the emanating sound:

- in acoustic emission the cause is the internal stress in the material,
- in impact methods the cause is an external impact on the tested object,
- in acoustic (vibration) monitoring the sound is caused by moving mechanical parts.

The advantages and disadvantages common to all three acoustic techniques are listed in Table 2.4.

Table 2.4: Advantages and disadvantages common to acoustic inspection methods.

Advantages	Disadvantages
• can evaluate entire structure during a single test, • defects in hard-to-get-to areas can be detected (e.g., in insulated vessels, partially covered, submerged structures), • inspection can be done during use of the inspected object (in-service testing), • acoustic signature can be used to determine type of defect.	• not all problem areas may be detected, use of other techniques may be necessary to complement acoustic methods, • acoustic signals may be obscured by many factors like geometry, materials, background noise, • no standardised interpretation methods.

Acoustic emission. In acoustic emission testing [Spanner et al., 1987] the acoustic energy that is detected is released from within the test object, rather than being supplied by an NDT method. The occurrence of an emission is effectuated by applying external load (stress). Metals subjected to stress and deformation can emit different sounds depending on prior heat treatment and the resultant internal crystalline structure. First applications of acoustic emission were studies of metal specimens during heat treatment and stress (for example, relation between stress-strain and the frequencies of the emitted sound).

The mechanism of the emissions is such that as the load is applied the sound emissions occur only once (irreversibility). This phenomenon is called the Keiser effect and in practice it means that inspection can be carried out only once with a given maximum load; to obtain new emissions either the load has to be increased or it is necessary to wait a longer period of time for the internal stresses to ease-out. This means that acoustic emission inspections have to be carefully planned as often they cannot be repeated (load cannot be increased beyond a certain maximum). Possible uses of acoustic emission technique are:

- detecting crack growth (a dynamic process),
- monitoring structural integrity of, for example, pressure vessels (also on-line),
- in-process (spot) weld monitoring,
- leak detection and location.

Acoustic emission examination is nondirectional (sources emit spherical waves). The emissions are registered usually using piezoelectric sensors. Monitored frequencies range from the audible range to 50MHz. A basic inspection system consists of a sensor and oscilloscope, but many systems for

acoustic emission testing are microcomputer based, mostly multichannel (4 to 128 channels). These multisensor systems can determine 1-, 2- or 3-D location of the detected emission events (for example crack initiation in a pressure vessel) by means of triangulation of signals from several sensors.

Apart from determining the occurrence and location of the emissions (emission events), the individual emissions can also be characterised. Usually, a single emission has the form of a burst with a short amplitude rise time and a longer decay time. A number of parameters can be used to characterise an emission event including:

- Emission counts — number of times a burst-signal wave crosses some amplitude threshold (signal ringing). Correlations can be established between the number of counts and various fracture mechanic parameters.
- Acoustic event energy and signal peak amplitude — these can be correlated to the mechanical energy in the emission event, and may be an indication for the amount of stress and deformation.
- Analysis of a number of parameters describing the event waveform (signal characteristics) with the goal of relating them to the characteristics of the emission sources. One can use, for example, statistical parameters of the signal (deviation, skewness, kurtosis), rise and fall time, various parameters from the frequency domain (for example, power in selected frequency bands), etc. If the real characteristics of the event sources are known then pattern recognition techniques can be used to construct signal classifiers, which can then be applied to classify data from unknown emission events.

Acoustic impact technique. In the acoustic impact technique a stress pulse is introduced into the tested object by a mechanical impact. Objects of smaller size and a higher modulus of elasticity will usually resonate with eigen frequencies. In larger objects or objects with low modulus of elasticity the impact will cause a wave to propagate in the object. Both the resonance frequency and the way the wave propagates will be primarily determined by the form and material of the object, but will also be effected by, for instance, internal defects. By monitoring the specimen acoustically after the impact one can detect these deviations as they will cause changes in the form of the signal.

The advantage of this technique is that a single measurement can be enough to inspect the whole structure or a large part of it. The disadvantage is low specificity of the acquired data. The technique may detect presence of abnormalities but characterisation of these abnormalities may be difficult. The typical uses are:

- monitoring of structural integrity,
- inspection of partially inaccessible objects (pipes, poles),
- testing bonded (e.g., glued) structures to determine quality of bonding, etc.

The received data is usually analysed in the frequency spectrum. The simplest way of interpretation is the determination of the frequency of the maximum amplitude peak. It gives the indication of the resonance frequency and this may indicate, for instance, a distance to a delamination in a composite layered material. A more complex analysis method may use pattern recognition techniques which relate the frequency-spectrum parameters to various defect types and characteristics, see for example, [Pratt & Sansalone, 1991] where an application to inspecting concrete structures is described.

Acoustic (vibration) monitoring. Mechanical and hydraulic systems with moving parts usually emit sound during operation. Under stable conditions a system causes a stable sound pattern with a cycle period being the result of superposition of all vibration frequencies. When the parameters of the system change, either due to working conditions and setpoints or due to a defect, then the sound pattern will also change. The pattern can be analysed to find the cause of the change.

This technique of defect detection will usually be used to monitor systems with relatively stable running conditions, so that the changes in the emitted sound pattern will be usually caused by some anomalies and not by a change in the operating conditions. One can use acoustic monitoring for on-line monitoring of manufacturing tools, turbines, bearings, etc. Analysis of the emitted sound pattern can also be used for inspection of pipes carrying fluids (detection of leaks, cavitation), high-voltage circuit breakers, etc.

The advantage of this technique is that one can inspect the whole structure at once, and that the inspection can be done without taking the system out of production. A disadvantage is the difficulty in the analysis of the measured acoustic signature. Sometimes it is enough to analyse the frequency spectrum in a similar way as described in the two previous sections. However, as already mentioned the acoustic emissions of some machinery are characterised by cycles and, therefore, must be analysed in the time-frequency domain. This can be done, for example, using Wigner-Ville distribution [Flandrin, 1988] or Short-Time Fourier Transform.

2.2.4 Radiation techniques

Radiation techniques make use of penetrating radiation to make images of internal structure of objects. The two types of radiation usually used for this purpose are X-rays and gamma radiation (from radioactive isotopes). When passing through an object, the radiation gets attenuated differently depending on the thickness and the material type. The amount of radiation that has passed through an object can be recorded on a photographic film. In recent years video imaging (analog or digital) of radiographic images has bee introduced. Such systems use either a converter plate that converts X-ray radiation to visible light or a CCD array directly sensitive to X-rays. The advantages and disadvantages of radiography are listed in Table 2.5.

Table 2.5: Advantages and disadvantages of radiography.

Advantages	Disadvantages
• high resolution, • can inspect thick objects, • a record of the inspection is always available (if film is used), • easy to interpret image, length and width of defects easy to determine.	• radiation hazard, • radiation source can be difficult to manipulate (X-ray sources can be large), • access to both sides of the object is needed, • not well suited for some complex geometries, • some types of defects difficult to detect (e.g., tight planar cracks), • depth of defects difficult to estimate (from a single image).

The differences between the X-ray and gamma techniques lie mainly in the difficulty in handling the radioactive gamma source, but this is compensated by the strength of gamma radiation, making the gamma technique suitable for inspection of thick pieces.

Inhomogeneities or defects larger than the system resolution should in principle be visible on the film, unless they are obscured or thin and located perpendicular to the beam. The problem in detecting some types of defects can be solved as follows:

- very small defects can be detected using microfocus X-ray tubes or image processing, e.g., deconvolution,
- defects located at a difficult angle can be detected by using several projections (views) of the object.

Special imaging techniques can be used like stereography using two radiation-source positions exposing either the same film or two different films. When video imaging is used, the position of the camera can be interactively varied, this way making it possible to visualize the 3-D location of the

elements in the image. The most sophisticated imaging technique is X-ray tomography which from a number of projections (can be as low as 15 [Hammar, 1998]) of the same object can reconstruct the full 3-D image.

Because, generally, automatic interpretation of images is complex, systems for automatic interpretation of radiographic images are rare. Automatic interpretation of the data is, for instance, used in the in-production inspection of high quality parts. There, a computer system can be used to compare the images of inspected parts to an image of a reference part.

2.2.5 Thermal techniques

Thermal techniques make use of heat sensors or infrared cameras to image the temperature distribution on the surface of inspected objects and structures. There are two main types of thermal inspection techniques:

- *Static* — In static methods inspection is done using images depicting temperature distribution at some equilibrium state.
- *Transient* — Transient methods examine the heat dissipation profile or pattern in the inspected object as a function of time. These methods heat the object using flash quartz lighting or a spot or line laser and observe the object as it cools down when the heat source is removed. Presence of zones with different heat dissipation properties influences the speed at which the surface cools and these zones become visible as differences in the surface temperature (after some time the temperature again will stabilize and without other heat sources will be the same over the whole surface). The transient techniques are, in general, more sensitive than static techniques.

The advantages and disadvantages of thermal techniques are listed in Table 2.6

Table 2.6: Advantages and disadvantages of thermal techniques.

Advantages	Disadvantages
• non-contact, • speed of inspection — large areas can be inspected quickly, • ease of deployment.	• quantitative results can be inaccurate, • sensitivity to surface artefacts of the tested specimen, • low penetration depth.

In static images the interpretation of the images is easy, and is usually confined to locating abnormally hot spots. In transient techniques the interpretation can be done by measuring the dissipation profile (this requires a sequence of measurements) or by capturing the pattern of temperature distribution at some exactly specified time after the heating pulse. Apart from pulse thermography also modulated thermography can be used with a modulated heat source. In case of modulated thermography the dissipation of heat is often treated as if it was a wave phenomenon. When this approach is used, one can determine the phase of the moving "thermal wave" which may provide additional information about the tested object.

Thermal techniques can be used to inspect, for example:

- in electronics, to localise faulty components on PCB boards,
- detect damaged coatings and insulations,
- determine thickness profiles of objects,
- spotting disbonds, delaminations, areas of porosity, and other defects inside composites,
- surface cracks can be spotted when the surface is thermically scanned with a laser (areas near the cracks have different heat dissipation characteristics),
- corrosion on the invisible side of the inspected object can be detected.

Static thermal images can in principle be analysed without help of computers. The transient technique requires use of a computer for precise data analysis. Computers can be used for the following tasks:

- visualisation and measurements in thermal images,
- compensation for nonuniform heating and contrast enhancement,
- calculation of phase images in modulated and pulsed thermography.

2.3 Quality of NDT

Every NDT inspection is associated with some cost, therefore, performing an inspection is fully justified only if its benefits outweigh the cost.[1] The cost of inspection is relatively simple to determine — roughly as the cost of the production halt plus depreciation of the inspection equipment and man-hours of the inspection team. Estimation of the benefit is not that easy. It requires knowledge of factors for which precise quantitative information is not always available. One of the factors is the estimated risk of continuing use of a product having some (possibly unknown) number of defects and the expected benefit of detecting some of them. The exact capabilities of the technique to be used are another such factor.

This section attempts to show that the determining factor in making decisions with respect to NDT inspection is the expected cost/benefit ratio. This is in turn determined not only by the inspection technique used but also by many other factors — these are discussed next. The discussion is based on [Wall & Wedgwood, 1994, 1998a; Wall, 1997].

2.3.1 Inspection performance

The benefit of inspection is largely determined by its performance. Inspection performance is influenced mainly by the four factors: sensitivity, speed, coverage, and reliability. These four factors are strongly interrelated, making optimisation of the performance a complex task.

Sensitivity. Sensitivity is defined by the minimum defect size that can be detected consistently. The required sensitivity depends heavily on the type of object inspected and on the goal of the inspection. Apart from the sensitivity of detection, the accuracy of defect characterisation is an important performance factor.

Speed. Speed of the inspection is determined not only by the time it takes to do a scan but also by the time it takes to interpret the data. However, often, during the scan time the inspected object cannot be used for its main purpose (for instance, a heat exchanger is taken out of production) and, therefore, different cost can be associated with the time for the scan and the time for the data interpretation.

Generally, the scan time is most significantly influenced by the inspection technique used (for instance, EC inspection is faster than penetrant methods). However, there is often a relation between the speed of the technique and its sensitivity, therefore, often a compromise between the two has to be made. A way to increase the speed is to reduce the coverage, by for example, doing ultrasonic measurements on a less dense grid. This reduces reliability, but also significantly reduces the cost, so that in the final calculation a reduced coverage may be a good choice.

A way to increase the scan speed without reducing the sensitivity is to use multiple probes. This increases the cost of the inspection equipment. Besides, handling multiple probes may be more difficult and the inspection personnel may be incapable of analysing the larger amount of data directly during the inspection (on-line).

1. Some inspections are prescribed by rules and regulations and have to be done irrespective of any cost/benefit calculations. Still, many regulations leave enough freedom for optimising the inspection method as far as the benefit and cost are concerned.

Coverage. Inspection coverage can be defined as percentage of the structure inspected. The main obstacle to achieve the 100% coverage is the geometry or location of the inspected object, and the capabilities of the technique used. For example, some NDT techniques can only scan a surface or a surface layer of the object. As already mentioned, coverage directly influences the speed of inspection and the reliability of inspection.

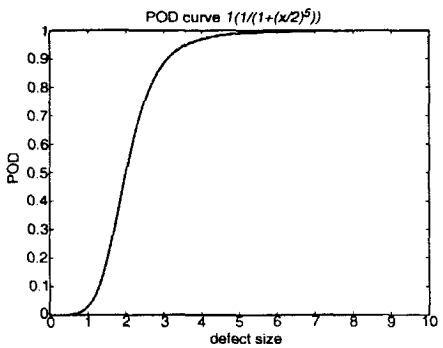
Reliability. Reliability of inspection is usually defined by two numbers: probability of detection (POD) and probability of false indication (PFI). The meaning of these parameters is discussed in the next section.

2.3.2 POD and PFI

POD (probability of detection) is defined as the number of defects detected divided by the number of defects (of the size/type which should have been detected) present in the inspected object. POD is usually determined using so-called "blind" tests, where a piece with known defects is inspected by an inspector who does not know the number, type and location of the defects. Because the number of defects is given and their size is known, one can calculate approximate probability of detection for each size group. Through these size/POD pairs a curve is fitted which relates the defect size to the POD, like in Figure 2.7. Experimental study of POD (e.g., tests done by Nordtest, and PISC I, II, and III trials [Bieth et al., 1998]) is quite expensive, but it gives realistic results as it includes also human factors, which are missed by theoretical modelling of POD based on the physical models for the inspection technique [Wall & Wedgwood, 1998b].

The POD depends on many factors like, for example:

- defect characteristics (type, shape, size, orientation),
- structural geometry of the inspected object,
- defect location (depth, nearness to structural elements),
- material properties (e.g., grain structure),
- accessibility of the inspected object, surface (shape, surface condition),
- NDT technique (its capabilities),
- NDT procedure,
- data processing and presentation,
- skill of the operator (data interpretation).



$$y = 1 - \frac{1}{1 + \left(\frac{x}{b}\right)^a}$$

Figure 2.7: POD curve and the corresponding formula as suggested by Nordtest [1998]. Another, often used, function to fit POD data is the Weibull cumulative distribution function.

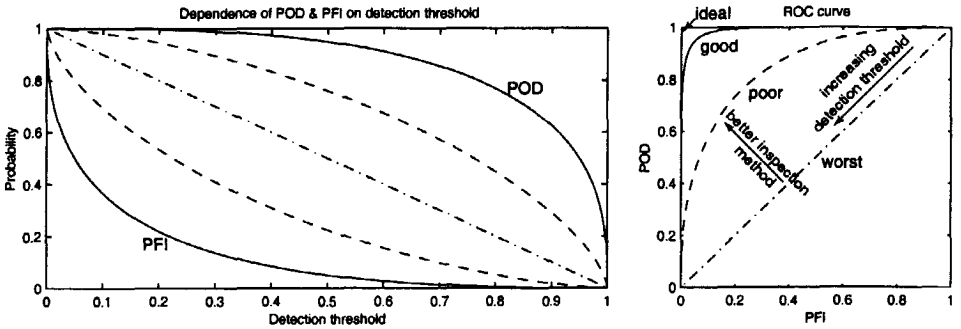


Figure 2.8: Changes of POD and PFI depending on the detection threshold, and the corresponding ROC curve.

The POD alone is not enough to describe the reliability of detection. It is easy to report any indications of the inspection system as defects. This increases the POD, but many of the reported defects will be no defects at all. This would be a problem because usually all the reported defects are subject to either additional inspection or immediate repair. This means that false-calls cost money. Therefore, full specification of the reliability includes also specification of the probability of such false indications (PFI).

While, POD is relatively widely used, use of PFI is not common.[1] Usually, one assumes that the PFI is acceptably low, but does not report it. Additionally PFI is not unanimously defined and it can be designated using other names as well (e.g., POFA – probability of false alarm). The ASME and Nordtest codes define PFI as *"the probability of false defect indication per size unit of the inspected object."* Calculation of PFI is easy for discrete objects, as the ratio of good but rejected objects to the total number of inspected objects. Instead of PFI one can also use false-call ratio (FCR) defined as the ratio of wrongly reported defects to all the reported defects (can also be expressed as function of defect size). FCR is easy to determine if all the reported defects will be checked after the inspection.

The dependence between POD and PFI is illustrated in Figure 2.8. In many NDT systems there is an explicit threshold which determines when an indication is considered a defect, see for example [Chiou et al., 1993]. Setting this threshold high will reduce the POD. In extreme case nothing will be reported — no defects and thus also no false indications. When the threshold is lowered smaller and smaller defects will be reported, but also some of these reports will be false indications. A detection threshold equal 0.0 is never used, in fact, it would mean that the whole inspected piece is considered defect before the inspection is started.

The dependence of POD and PFI on the detection threshold can also be plotted as a single graphic called ROC curve [Okure & Peshkin, 1995]. ROC stands for Receiver Operating Characteristics and comes originally from the signal detection theory (statistical decision theory) [Van Trees, 1968]. ROC curves are widely used in medicine to evaluate diagnosis methods. Points along the curve correspond to various values of POD/PFI pairs depending on the value of the detection threshold. The form of the curve gives an idea about the goodness of the detection system (irrespective of the later-to-be-chosen detection threshold). A random classification system gives a straight ROC curve. The better the system the closer the curve moves to the POD=1, PFI=0 point. The size of the area between the curve and the diagonal is often used as an estimate of the reliability of the system.

1. In medical diagnosis, for instance based on X-ray images, one pays much more attention to the falls-calls, possibly because they are more apparent, and also because they can significantly reduce confidence in the diagnosis. In this respect, developments in the POD/PFI analysis in NDT trail these in the medicine.

Parameters affecting the shape of the ROC curve are:

- information available — is it sufficient for good separation of defect and non-defect indications,
- noise in the signal — more noise means more overlap in the distribution of the parameters describing the defect and non-defect indications.
- the decision criterion — it does not have to be a simple threshold (a linear discriminator).

This suggests several ways to improve the reliability of inspection, like: obtaining more information (e.g., use more frequencies in EC inspection), reducing noise in the signal, and/or use of a more sophisticated signal interpretation method.

2.3.3 Cost aspects

For the purpose of decision making, one often needs to determine the value of a certain way of doing the inspection. A convenient way to express that value is simply in terms of money. The inspection is then justified if the expected benefit is higher than the expected cost of the inspection. The positive difference between the two is called the added value of the inspection.

The benefit. The benefit of inspection is the avoided cost of failure that might be caused by the undetected defects. The cost of failure is not always easy to ascertain and is often determined not by the primary damage but by the resulting damage. A damage caused by a leak in a water pipeline will be much smaller than potential consequences of a leak in a gas pipeline. Bulk-cargo carriers and oil tankers are another example; in practice nobody cares if a bulk-cargo carrier sinks, but oil-spills have almost always serious consequences. Damage cost estimation is especially difficult when human lives are at risk.[1]

Generally, the cost of damage is not proportional to the number of the defects (it is quite possible that a single defect can cause a failure that might destroy the whole structure), but the probability of the failure is generally proportional to the number of the defects. Probability of failure can be expressed as a function of time, the idea of which is shown in Figure 2.9. This is a rising function because, generally, population and size of defects grow with time. If, during an inspection, some of

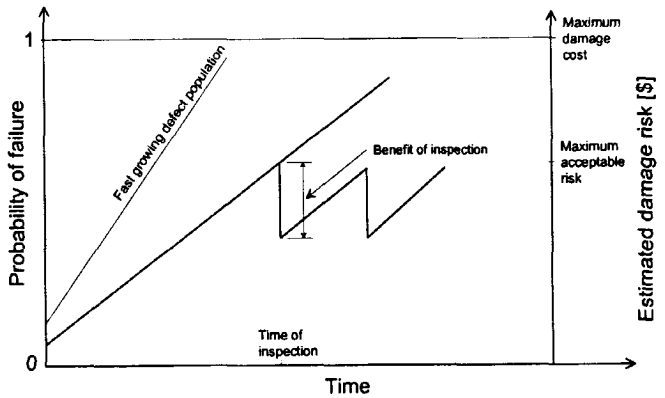


Figure 2.9: Probability of failure and estimated risk as function of time. Carrying out an inspection (and taking preventive actions based on it) reduces damage risk. This damage-risk reduction can be considered as the benefit of that single inspection.

1. For example, after the Union Carbide disaster in Bhopal, India (Dec. 1984), the compensation paid by the company amounted to about $30,000 per dead person, while it is said that British Rail handle the sum of 1 million £ per dead person in their risk analyses.

the defects are detected and the structure is repaired then the probability of damage will decrease, which, knowing the estimated cost of failure, can be translated into the benefit of inspection. In complex structures, with various modes of failure, having various probabilities and different related cost of failure, the use of a global probability of failure may be inappropriate. From the point of view of decision making, an estimated (in terms of probabilistic expectation) cost of failure is an easier to interpret value.

The benefit of inspection is largely proportional to the percentage of defects detected, this, in turn, it is clearly related to the probability of detection (POD) discussed in the previous section. The further in time the inspection is done the higher the population of the defects and the bigger they are; this means that for a given POD curve a larger percentage of defects will be detected, apparently leading to a greater benefit of inspection, but this also means that the structure would have to be operated at a higher risk, which is usually not allowed by existing regulations.

The cost. The *cost* of inspection is determined by the following factors:

- The cost of the out-of-production time of the inspected object – tanks may have to be emptied, chemical installations have to be shutdown, etc.
- The cost of actually *doing* the inspection. This is, in turn, determined by a number of factors:
  - The cost of equipment – depending on the technique used, level of automation, etc.
  - The cost of personnel – depending on their number and qualifications.
  - The inspection coverage – the more measurements have to be done, the more expensive the inspection is.
- The cost of repair of the detected defects, like making new welds, replacing parts, plugging pipes in a heat exchanger, etc. Cost of repair is not always linearly proportional to the number of defects, especially when whole parts have to be exchanged. Knowing precisely the size of the defects, one can decide better what repair actions should be taken.
- The cost of intervention in case of false-calls. It can be smaller than the cost of repair if, for example, additional more precise tests are done and the repair can be avoided.

The added value. The added value V of a single inspection can be expressed as:

$$V = B - C_R - C_F - C_I$$

where: B is the benefit of detecting defects, C_R is the cost of repair of detected defects, C_F is the cost of false calls, and C_I is the total cost of inspection (includes out-of-production cost). This approach does not take the level of damage risk into account, or rather it assumes that the maximum allowed damage risk is a given variable. In other words, a decision is made about, for instance, what inspection technique should be used, and to a lesser extent when the inspection is to be made.

One could also take the damage risk into the equation and define the optimal inspection strategy (that is the inspection technique and schedule) as such which minimizes over the whole product lifetime, the sum of damage risk and inspection cost. But that is a more difficult approach.

Maximising the value of inspection. Maximising the value of inspection involves many interrelated factors making the whole process a difficult one. Optimising the inspection with respect to one factor may influence other factors, possibly annulling any gains. Two additional difficulties are:

- Many factors need to be estimated and for some of them the estimates are only approximate, like for damage risk, human factors, etc. This means that a single result is not really conclusive and has to be accompanied by sensitivity analysis, i.e., determination how the cost/benefit model used is sensitive to changes in the (estimated) parameters.
- Many parameters are constrained by rules and regulations, on the one hand making it difficult to obtain really optimal results, but, on the other hand, simplifying decision making, especially in situations where high risk is involved.

In Table 2.7 several ways to maximize the value of the inspection by changing single factors are listed. Some of the possible negative influences on the other factors are also mentioned.

Fitness for Purpose (FFP). A comprehensive approach to the reduction of the cost of inspection is the so-called Fitness for Purpose (FFP) approach. A structure can be considered fit for purpose when it complies with all operational and safety requirements during its full lifetime. One of the objectives of FFP is prevention of unnecessary rejections or repairs. This means that FFP approach

Table 2.7: Several ways to maximise the added value of an inspection.

Component	How to optimise	Possible negative consequences
benefit B	**increase benefit of inspection**	
	inspect areas where most dangerous defects are expected more carefully (better coverage, better NDT technique)	may increase cost somewhat (preparing and doing inspection, requires careful analysis)
	increase probability of detection (POD)	
	increase coverage – uniformly over the whole object, or do sampled inspection	increases time and cost
	use a better NDT technique	increases cost and maybe time
	improve data processing and interpretation	increases cost (investment which should pay back)
cost of inspection C_I	**decrease out-of-production cost**	
	do inspection faster, automate it	may reduce POD, increase cost of doing inspection
	separate scanning from interpretation	requires more expensive equipment, less feedback from actually doing the inspection
	reduce need of preparation for inspection, use technique which allows for in-production inspection	may reduce POD, the technique itself may be more expensive
	decrease cost of doing inspection	
	decrease coverage	may reduce POD
	use a cheaper technique	may reduce POD, increase PFI
	use faster inspection technique	may reduce POD, may require off-line data analysis
	automate inspection and data interpretation to make it *faster, requiring less personnel*	large initial investment, may be cost effective only over a longer time
	design the product for inspectability (e.g., easy access for automated NDT inspection)	possibly higher product cost
cost of false calls C_F	**decrease PFI**	
	use a better NDT technique	increases cost and maybe time
	use better processing and (automatic) interpretation	initial investment cost
cost of repair C_R	**decrease repair cost**	
	provide accurate characterisation (sizing), so that repair can be put off if not immediately necessary	may increase cost of inspection, requires knowledge about failure mechanics of the inspected structure
	use a combination of methods, one with good POD, another with accurate sizing	may increase cost of inspection
	monitor defects instead of immediately repairing them	requires precise (characterisation & localisation) technique (more expensive) and periodic inspection regime
	design product for controlled failure – repair only biggest defects	product more expensive, not always admissible

allows for presence of imperfections which do not lead to failure. To achieve this, two components are necessary:

- Analysis of failure (fracture) mechanics. Relation between the defects present in structure and the probability of failure has to be found.
- Analysis of the capabilities of candidate NDT inspection techniques. This includes mainly POD, PFI, and sizing accuracy.

When these two are known one can determine the most optimal technique to be used, the decision threshold for the repairs, and the inspection interval. FFP approach to inspection makes sense especially if it spans the whole lifetime of the structure as only then the maximum economic gains can be expected. The FFP approach is not widely used because the required problem analysis involves significant cost. Also, some users may be uncomfortable with explicitly allowing for an acceptable probability of failure (though it is present in any inspection due to a non-100% POD). To reduce the cost of analysing each inspection problem individually, some generally applicable FFP guidelines are being or have been developed like, for example, recommendations for the inspection of welds described in [BSI, 1991].

2.4 Automating NDT inspection

One way to increase the added value of the inspection, listed in Table 2.7, is automating the inspection. Automated inspection can contribute to the reduction of the inspection time, reduction of the man-hours of the qualified personnel for data interpretation, increase POD and reduce PFI (thus also improve reliability) by more sophisticated and more consistent processing, and provide more valuable inspection results due to more accurate defect characterisation. Figure 2.10 shows the two ways of automating the inspection: automated scanning and automated interpretation.

2.4.1 Automated scanning

Automated scanning applies to the ways of positioning and moving a sensor (or sensors) over the scanned path, surface or volume. Scanning may be done by moving the sensor (and the source) using, for example, a push-puller (1-D scans), a 2- or 3-D scanning system (for flat or curved surfaces), or a robot (for scans along a predefined path [Vesth et al., 1998]). Scanning can also be done by using array sensors, for example, an ultrasonic array sensor may be made to move its focus point by adjusting the pulse-time delays to its elements.

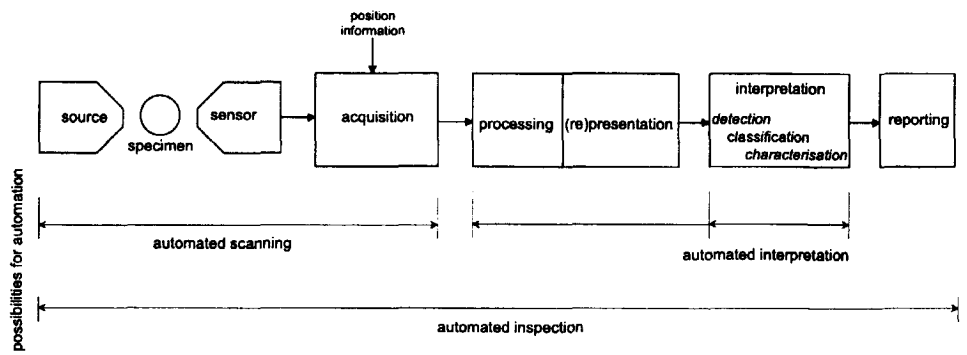


Figure 2.10: Ways of automating the NDT inspection chain. Automated interpretation may also include special preprocessing.

The main benefits of automatic scanning are:

- *Increase in the speed of the inspection* — This will be the case if the other elements of the inspection chain can keep up with the faster stream of data. The problem may be the on-line interpretation. If the data is recorded then interpretation can be done off-line, and the faster inspection can contribute to a lower total cost due to a shorter out-of-production time.
- *Increase in the positioning accuracy* — An automatic scanning system may be made to follow a precise path, so that the scanning beam reaches exactly the area that has to be inspected.
- *Availability of the positioning data* — This allows to use more sophisticated processing techniques (2-D, 3-D imaging), and to monitor of the condition of the structure over a period of time (comparison of results from several consecutive inspections).

The use of automatic scanning has some disadvantages, like:

- *Higher cost* — The cost of an automatic scanning system itself is higher than that of a system where scanning is done by hand, but the overall economic balance will usually be in favour of the automatic system, unless the inspection is incidental, concerns a small area, etc. (see next point).
- *Difficulty of handling* — An automatic scanning system may be difficult to use on objects with difficult geometries, in cramped spaces, etc.

2.4.2 Automating NDT data interpretation

The use of automatic scanning, as well as use of new inspection techniques, for instance multiple sensors, leads to an increased amount of data. Often the operators cannot keep up with analysing this stream of data. This creates a need for reliable automatic or automated data interpretation techniques.

The use of automatic interpretation techniques serves, first of all, to increase the speed of the data interpretation. However, as already mentioned in this chapter, increases in the reliability, consistency, and reproducibility of the inspection are also important goals. Automatic interpretation may help in precise defect characterisation which is important for FFP. Also, in some situations, automated interpretation can help to interpret complex data because they are capable of analysing more data sources simultaneously than could be handled by an operator.

Section 2.1 has already presented the three components of interpretation: the detection, the classification, and the characterisation. This section discusses how these three components can be automated.

Detection. Detection of the possible defect indications in the signal, in an image, etc., is the basis of data interpretation. In fact, a reliable automatic defect detection system may significantly reduce the cost of inspection because it will let the operator concentrate on the defect classification and characterisation. Usually, most of the inspected structure is without defects. If the operator has to spent hours watching data with no defect indications then there is a high probability that he may miss defect indications when these occur in the signal.

In some applications the detection of possible signals from defects can be as trivial as simple thresholding. However, this method fails when an uncorrected offset or drift is present in the signal. Drift or offset correction though relatively easy for an operator may be quite difficult for automatic systems because it often requires analysis of the whole of the signal. Moreover, indications from some defect types may look very similar to a drift. Often, an automatic system may have to recognise various portions of the signal (like pipe entry and U-bent in the case of heat-exchanger pipe inspection) to apply the correct detection method to that part of the signal.

Another problem in defect detection is noise. This is especially the case for signal from small defects which may have amplitude comparable to the amplitude of the noise signal. Noise can be partly removed by use of various filters. This is, however, difficult if the noise has the same frequency components as the signal from the defects. In such situations use of simple filters may lead to

the distortion of the defect signal making its classification and characterisation difficult. If this is the case, one may have to use methods of noise removal based on noise modelling [Benoist et al., 1995].

Detection may be especially difficult in images, for example, B-scans, C-scans and X-ray images. In such situations, detection may require use of image recognition techniques. One can use techniques like template matching (where the templates used correspond to images of expected defects) or various image segmentation techniques. The detection problem becomes even more difficult if the image also shows the construction characteristics of the inspected objects, like holes, profile changes, etc.

Classification. Proper classification is necessary in order to distinguish between indications from defects and non-defects and also to distinguish between the various defect types as these may require different characterisation techniques and may have different contributions to the risk of failure.

Classification of the signals is usually the main area of expertise of the NDT inspector. It is usually done based on a large number of parameters which do not lend themselves to simple algorithmic treatment. Therefore, when the problem of automatic classification has to be solved one often looks for a solution in the pattern recognition and/or knowledge-based techniques [Wielinga et al., 1994]. Still, in some situations the problem of classification can be solved by use of appropriate data processing based on the model of the inspection. This way, for example, influences of construction features can sometimes be removed from the data. One of the examples is signal mixing in the eddy-current technique described in Section 2.2.1. While the interpretation of the compound baffle and defect signals is difficult [Lord & Satish, 1984], the signal classification after mixing is significantly easier. The automatic classification is also easier or achieves better results if the noise has been removed from the data, or other techniques to improve the clarity of the data have been used, like deconvolution of the sensor characteristics [Downs & Peterson, 1997].

The techniques for the NDT data interpretation described in chapter 5 concentrate mainly on data classification.

Characterisation. Defect characterisation should ideally be done based on the models relating parameters of the defects to the parameters of the measured signal. However, as already mentioned in Section 2.1 most of the characterisation is done based on the calibration results. Usually, calibration results relate, by means of a calibration curve, a single parameter of the defect signal to a single parameter of the defect. Use of more-dimensional calibration curves/surfaces (taking several parameters of the signal into account) or of several calibration curves (corresponding to several defect types) could improve the precision of the defect characterisation. However, this is rarely or never done because of two reasons. First, it would increase the cost of the calibration. Second, it would be difficult to handle by the inspector. A single parameter can be easily displayed on a display, while multiple or multi-parameter calibration curves would be difficult to visualise. For an automatic characterisation system use of multiple-parameter calibration curves is not a problem.

Sometimes characterisation is done based on the measurements in the images (for example X-ray). This can be done by the operator, but an automatic system can do the same faster and more precisely. Automatic systems may also make use of the available models for the inspection and by means of inversion determine the parameters of the defects [Faur et al., 1997].

One may observe that a better inspection technique and better data (pre)processing may simplify the task of automatic interpretation. When building an automatic inspection system one has, therefore, to determine which part of the system will bring the most gain.

Fully automatic inspection is essentially feasible only as a part of a manufacturing process, for instance, inspection of wires, rails, point welds. There, the inspection conditions are well defined and the cost of mistakes is relatively small. On-site inspection of, for example, petrochemical instal-

lations, pipelines involves too many unknowns to be done fully automatically. Therefore, when building an automated inspection system one can (or has to) take into account the presence of an operator and design the system for assistance of the operator and not his replacement.

The last, but not least, advantage of the use of automated inspection (scanning and interpretation) techniques is better quality of reports. The location of the defects is precisely specified and the defect characterisation is free from the operator bias. Because the reports are more consistent, analysis of trends between inspections becomes feasible. Based on the result of the trend analysis it is, for instance, possible to adjust the coverage of the inspection or the period between the inspections and further reduce the cost of the inspection.

2.4.3 Typical existing automated inspection systems

Automatic scanning equipment is already quite common in commercially available NDT systems. The same is true about digital acquisition and storage of data. Many systems also provide means of automatic detection of possible defect indications, for example, by means of thresholding or peak detection. Some systems, like for example ECSF1 for EC inspection from Amtak, Inc., may provide classifiers (statistical or ANN) which can be trained to classify the detected signals, however, this functionality is rarely, if ever, used in normal field inspections.

Many systems, for example most of these used for EC inspection, provide means of automatic signal characterisation without prior classification. The characterisation can be done using calibration curves (the systems usually provide means for their easy construction). However, the same characterisation procedure is usually applied to all detected signals, some of which are not defects. This means that an operator has to check all the output of the system; he can accept, reject or modify the computer analyses. Examples of such systems for eddy-current inspection are MAD 4D from etc. Eddy Current Technology and eddyMax® from TMT. They are widely used and save much work.

A component almost always present in any modern computerised NDT inspection system is reporting software which stores the result of the inspection in a database and can be used to prepare graphic reports which give a good overview of the condition of the inspected object.

2.5 References

Adler, L., Jungman, A., Nagy, P., and Rose, J. (1991) "Waveform and data analysis techniques", *Nondestructive Testing Handbook: Volume 7, Ultrasonic Testing*, (2nd ed.), A.S.Birks, R.E. Green, and P. McIntire (eds.), ASNT, pp. 131–185.

Benoist, B., Gaillard, P., Pigeon, M., Morizetmahoudeaux, P. (1995) "Expert system for the characterization of defect signals in steam generator tubes", *Engineering Applications of Artificial Intelligence*, Vol. 8, No. 3, June, pp. 309–318.

Bieth, M., Birac, C., Comby, R., and Maciga, G. (1998) "In-Service Inspection Capability of Steam Generator Tubes", *7thECNDT Proceedings*, Vol. 2, pp. 2029–2036.

Birks, A.S., Green, R.E. Jr., and McIntire, P. (eds.) (1991) *Nondestructive Testing Handbook, Volume 7, Ultrasonic Testing*, (2nd ed.), ASNT.

BSI (1991) *Guidance on methods for assessing the acceptability of flaws in fusion welded structures*, BSI PD 6493:1991.

Cecco, V.S., Van Drunen, G., and Sharp, F.L. (1983) *Eddy Current Testing, Vol. 1*, Atomic Energy of Canada Limited, Chalk River Nuclear Laboratories, Chalk River, Ontario.

Chiou, Ch.-P., Shmerr, L.W., and Thompson, R.B. (1993) "Ultrasonic flaw detection using neural network models and statistical analysis: Simulation studies", *Review of Progress in Quantitative Nondestructive Evaluation*, Vol. 12A, D.O. Thompson and D.E. Chimenti (eds.), Plenum Publishing, New York, pp. 789–795.

Downs, J. III and Peterson, M.L. (1997) "Theoretical, simulated, and experimental resolution enhancement of a transducer by deconvolution of the point spread function", *Review of Progress in Quantitative Nondestructive Evaluation*, Vol. 16A, D.O. Thompson and D.E. Chimenti (eds.), Plenum Publishing, New York, pp. 717-724.

Faur, M., Roy, O., Benoist, Ph., Oksman, J., and Morisseau, Ph. (1997) "An inverse method for cracks characterisation from ultrasonic bscan images", *Review of Progress in Quantitative Nondestructive Evaluation*, Vol. 16A, D.O. Thompson and D.E. Chimenti (eds.), Plenum Publishing, New York, pp. 67–74.

Flandrin, P. (1988) "Non-Destructive Evaluation in the Time-Frequency Domain by Means of the Wigner-Ville Distribution", *Signal Processing and Pattern Recognition in Nondestructive Evaluation of Materials*, C.H. Chen (ed.), Springer-Verlag, pp. 109–116.

Hagemaier, D.J. (1983) "Eddy current impedance plane analysis", *Materials Evaluation*, Vol. 41, February, pp. 211–218.

Hammar, L. (1998) "In-service Inspection by X-Ray Tomosynthesis, A New Method for Sizing IGSCC", *7th ECNDT Proceedings*, Vol. 2, p. 1517.

Krzywosz, K. and Dau, G. (1990) "Comparison of Electromagnetic Techniques for Nondestructive Inspection of Ferromagnetic Tubing", *Materials Evaluation*, Vol. 48, January, pp. 42–45.

Lord, W. and Satish, S.R. (1984) "Fourier descriptor classification of differential eddy current probe impedance plane trajectories", *Review of Progress in Quantitative Nondestructive Evaluation*, Vol. 3A, D.O. Thompson and D.E. Chimenti (eds.), Plenum Publishing, New York, pp. 589–603.

Monebhurrun, V., Duchene, B., Lesselier, D., and Zorgati, R. (1995) "Eddy current characterization of 3-D bounded defects in metal tubes using a wavefield integral formulation modeling", *Nondestructive Testing of Materials*, R. Colins et al. (eds.), IOS Press, pp. 195–202.

Nordtest (1998) *Guidelines for NDE Reliability Determination and Description*, NT TECHN REPORT 394.

Okure, M.A.E. and Peshkin, M.A. (1995) "Quantitative evaluation of neural networks for NDE applications using the ROC curve", *Review of Progress in Quantitative Nondestructive Evaluation*, Vol. 14B, D.O. Thompson and D.E. Chimenti (eds.), Plenum Publishing, New York, pp.2405–2412.

Pratt, D. and Sansalone, M. (1991) "The Use of a Neural Network for Automatic Impact-Echo Signal Interpretation", *Review of Progress in Quantitative Nondestructive Evaluation*, Vol. 10A, D.O. Thompson and D.E. Chimenti (eds.), Plenum Publishing, New York, pp. 667–674.

Smith, R.A. (1998) "Fine-tuning the eddy current detection of hidden first-layer corrosion in aircraft skins", *INSIGHT*, Vol. 40, No. 10, October, pp. 712–721

Spanner, J.C., Brown, A., Hay, D.R., Mustafa, V., Notvest, K., and Pollock, A. (1987) "Fundamentals of Acoustic Emission Testing", *Nondestructive Testing Handbook*, Vol. 5., R.K. Miller and P. McIntire (eds.), ASNT, pp. 11–44.

Stolte, J., Udpa, L., and Lord, W. (1988) "Multifrequency eddy current testing of steam generator tubes using optimal affine transformation", *Review of Progress in Quantitative Nondestructive Evaluation*, Vol. 7A, D.O. Thompson and D.E. Chimenti (eds.), Plenum Publishing, New York, pp. 821–829.

Van Trees, H.L. (1968) *Detection, Estimation, and Modulation Theory*, Part I, John Wiley & Sons, Inc.

Vesth, L., Hansen, P.B. and Damgaard Kristensen, W. (1998) "Robot System for Ultrasonic Examination of Complex Geometries", *7th ECNDT Conference*, Vol. 3, pp. 3128–3133.

Wall, M. (1997) "Modelling of inspection reliability", *Engineering Science and Education Journal*, April, pp. 63–72.

Wall, M. and Wedgwood, F.A. (1994) "NDT - Value for money", *INSIGHT*, Vol. 36, No. 10, October, pp. 782–790.

Wall, M. and Wedgwood, F.A. (1998a) "Economic Assessment of Inspection — The Inspection Value (IVM) Method", *7th ECNDT Conference*, Vol. 2, pp. 2053–2060.

Wall, M. and Wedgwood, F.A. (1998b) "Modelling of NDT Reliability (POD) an Applying Corrections for Human Factors", *7th ECNDT Conference*, Vol. 2, pp. 2108–2115.

Wenk, S.A., McMaster, R.C., and McIntire, P. (eds.) (1987) *Choosing NDT: Applications, Costs and Benefits of Nondestructive Testing in Your Quality Assurance Program*, ASNT.

Wielinga, T.S., Kerckhoffs, E.J.H., and Lorenz, M. (1994) "Towards the classification of round and planar weld defects with neural networks", *Neural Network World*, Vol. 4, pp. 465–478.

Zala, C.A., Barrodale, I., and McRae, K.I. (1988) "High Resolution Deconvolution of Ultrasonic Traces", *Signal Processing and Pattern Recognition in Nondestructive Evaluation of Materials*, C.H. Chen (ed.), Springer-Verlag, pp. 101–108.

Chapter 3

Artificial Intelligence

Are we [AI] science or engineering, analytic or synthetic, empirical or theoretical? The answer of course is, "yes."

— Randall Davis (1998) "What Are Intelligence? And Why?", AI Magazine, Vol. 19, No. 1, p. 94.

3.1 Understanding intelligence

3.1.1 What is intelligence?

Intelligence is a difficult to define concept. It is generally understood to refer to ability to learn, comprehend, think, reason, etc. It can have a very wide scope of meanings. Some equal being intelligent practically with being human, for example, the Turing test (discussed further in this chapter) is, in its original form, a test for humanness. On the other extreme, there is a popular use of the word intelligence to describe things like an intelligent terminal, smart card, intelligent (hardware) interface, etc.

Sometimes, especially when confronting human intelligence with machine intelligence, the handled definition of intelligence becomes tinted with a requirement of emotions. However, especially in the past (for instance, within the romanticism of the end of XVIII century), intelligence, and intellect in particular, were contrasted against emotion. Even today, the figure of Mr. Spock from the "Star Trek" series exemplifies someone intelligent but without human emotions. It is unclear whether consciousness, feelings, and emotions should be included in the definition of the intelligence.

Thus, it seems that intelligence is an elusive notion. Still, it is common to express opinions about someone being more intelligent or less intelligent, implying that intelligence can somehow be measured. A person's intelligence can usually be determined from observations of one's behaviour in various situations, that is why, in psychology, a large number of largely standardised tests have been developed which are supposed to measure intelligence. Such tests allow us to define "the intelligent" as such that receive high enough scores on intelligence tests. The behaviouristic approach seems to take away the problem of the *essence* of the intelligence (which we are unable to judge for non-humans) and concentrates of the outside, the measurable. This seems appropriate, as intelligence is usually manifested in actions — it is difficult to judge "static" intelligence. This definition of intelligence seems, at first, to be useful to determine if non-human subjects (e.g. computers) are intelligent. However, most of the intelligence tests are made with humans in mind, which is specially clear whenever time limits for performing a task are used.

On the other hand, while to a certain extent practical, such a behaviouristic definition is not fully satisfactory. When we are interested in intelligence we are usually as much interested in how the results (the intelligent behaviour) are achieved as in the results themselves. That looking at the results seems to be in many cases satisfactory is because we assume that everybody thinks in a similar way (the same way as we do). Judging the intelligence becomes more difficult when apart from different results also different ways of achieving these results come into play. Some of the problems which require intelligence to be solved by humans can also be solved by trying out all the possibilities by brute force, and such approach would generally not be called intelligent.

3.1.2 Artificial Intelligence

Artificial Intelligence (AI) is all about programming computers to perform tasks normally associated with intelligent behaviour. Classical AI programs have played games, proved theorems, discovered patterns in data, planned complex assembly sequences and so on. AI addresses issues related to the representation and the use of knowledge of human experts. It also provides means to model and to help us understand human intelligence.

The origins of the Artificial Intelligence date from the 1940s and the early 1950s. In 1940s first experiments were made with neuron simulators by Warren McCulloch and Walter Pitts and the work on dynamic memory was done by Donald Hebb. In the early 1950s, a symbolic interpretation of the working of human mind was developed by Alan Turing. From the beginning, the research in the AI has proceeded along two paths. The first one, associated with *cognitive science*, tried to use computers to explain the essence of intelligence. The other concentrated on the use of "intelligent" systems in practice. The goal of this branch, sometimes called *knowledge engineering*, is to build knowledge-based systems. Thus, some of those who work in the field of Artificial Intelligence are relatively unconcerned about whether the programs they devise mimic human cognitive functioning, while others have the explicit goal of simulating human cognition on the computer. Knowledge engineering borrows from the ideas developed by cognitive science and vice versa.

Both approaches stress the following properties of intelligent systems: ability to communicate, goal orientedness, knowledge, learning, ability to cope with errors and ambiguity. Cognitive science is also interested in the consciousness (self awareness) while for knowledge engineering this is relatively unimportant.

Turing test. One of the "fathers" of AI, Alan Turing was convinced that mind is a machine doing symbolic processing, and sooner or later a computer will be built which behaves like human.[1] Turing put forward the idea of an "imitation game" [Turing, 1950], in which a human being and a computer would be interrogated under conditions where the interrogator would not know which was which, the communication being entirely by textual messages. Turing argued that if the interrogator could not distinguish them by questioning, then it would be unreasonable not to call the computer intelligent. Thus the Turing test represents a behavioural approach to determining whether a system is intelligent.

By suggesting the "imitation game" Turing tried to simplify the problem of answering the question "Can machines think?". However, in his article, as if expecting the unsatisfactory character of the simplification, Turing discusses mainly the original broad question.

The original Turing test is flawed as it is a test for *human* intelligence, or rather, effectively, a test of how a computer can fool the interrogator. The example of the test given by Turing [1950, pp. 434–435] shows that the machine is not trying to imitate an intelligent way of thinking but necessarily, due to a way the test is constructed, a human way of thinking thus making mistakes in calculations and taking some time to deliver answers. The types of questions asked refer to calculation and playing chess (thought to require intelligence to be performed by humans) but also to poetry which essentially involves emotions and memories.

It might seem that it would be difficult to build a machine capable of passing the Turing test (Turing himself predicted that around year 2000 a machine would be built which would have a 70% chance of fooling the interrogator). However, as early as 1960s a relatively simple program called ELIZA, written by Joseph Weizenbaum at MIT during 1964-1966, passed, in opinion of some, the Turing test. The program ran a DOCTOR (a psychiatrist) script and it managed to fool many peo-

1. In his famous 1950 article *"Computing Machinery and Intelligence"* Turing writes (pp. 455–456): *Presumably the child-brain is something like a note-book as one buys from the stationers. Rather little mechanism and lots of blank sheets. ...Our hope is that there is so little mechanism in the child-brain that something like it can be easily programmed*

ple into thinking they had a conversation with a real psychiatrist. Weizenbaum was shocked at reactions to the program: psychiatrists thought it had potential, people unequivocally anthropomorphised, many thought it solved the natural language (NL) problem. This example shows how one can simulate intelligence/humanness without any real understanding.[1]

The Chinese Room [Searle, 1980]. Within Artificial Intelligence there have been two tendencies: so called *weak* and *strong AI*, and there has been a continuing argument between the supporters of either of them. Strong AI claims, among other things, that: mind *is* a computer and, vice versa, an appropriately programmed computer is a mind; a machine can literally be said to *understand* stories; to have a mind is to run the right computer program; all thinking is computation, in particular, *feelings* of conscious awareness are evoked merely by the carrying out of appropriate computations. Weak AI claims only that minds may be simulated using computers (thus we can achieve similar results but the way they were achieved gives us no exact insight in the working of the mind) and intelligence is not necessarily equivalent with consciousness. To strong AI, understanding is simply correct applying knowledge in practice and nothing more. To weak AI, understanding implies a full comprehension of the meaning of a concept and AI programs are incapable of it.

Searle tried to demonstrate that contrary to the claims of strong AI implementing a (computer) program is not sufficient for mentality. Searle illustrates this using an example of a room in which there are books describing how to process Chinese symbols. There is also somebody in the room who does not understand Chinese but who knows how to follow instructions in the books. If now somebody asks the person in the room a question in Chinese he is able to give a correct answer by using instructions from the books, but still he does not understand Chinese. Thus, according to Searle strong AI is false, and no computer program is capable of self-awareness (consciousness).

There have been various responses against the Chinese Room argument, for example, the *system argument* — that though the man doing the processing in the Chinese Room does not understand Chinese the systems as a whole does; and the *robot argument* — if the system is equipped with ways of sensing and interacting with the world it may have means to understand the meaning of symbols.

The Chinese Room argument draws attention to the fact that declarative or functional programs are not considered Artificial Intelligence. However, majority of AI systems are implemented on digital computers, using declarative programming. Though at a higher abstraction level they might seem intelligent, at the level of the declarative program there are certainly no "cognitive states" or any consciousness or awareness.

3.1.3 Historical AI — the two paradigms

Two paradigms have been present in the AI from the very beginning — the symbolic and the connectionist one.

Symbolic representation and reasoning. The symbolic paradigm is essentially based on the use of words to describe the world around us. People use symbols (words) to represent realities and abstractions. The symbols themselves can be considered as arbitrary in relation to what they can represent (mean). These symbols can be communicated and manipulated. Symbol manipulation is done using rules operating only on the symbols' shapes.

The symbolic paradigm has been worked out by Turing. Turing has developed the idea of automata which can process symbols using instructions also expressed as symbols. He has shown that the so called Turing Machine is capable of solving any computable problem, computation being interpretable symbol manipulation.

1. In 1990 a Loebner Prize (www.loebner.net) was established for a program passing the Turing test. What is characteristic of the submissions to the contest is that they are relatively simple programs that use a whole bag of tricks to fool the judges.

Symbolic AI refers to the classical view of the architecture of the mind (compare footnote on page 34). In this approach the mind is viewed as a process in which symbols are manipulated. Symbols are moved between memory stores such as long term and short term memory and are acted upon by an explicit set of rules in a particular sequence. The symbolic architecture approach has been widely applied and forms the basis of expert systems (see Section 3.2).

Connectionist approach. While symbolic AI concentrates its efforts on the presupposed way humans process the information, the connectionist AI deals with the human information-processing "hardware" – the brain – and tries to simulate it. The precursors of connectionist AI were Warren McCulloch and Walter Pitts who in the beginning of 1940s worked on neuron simulators. Further research concentrated on making more and more complex networks of neurons and on finding the best ways to train them.

In a connectionist architecture no direct meaning can be assigned to the internal states of neurons and their connections. Therefore, one speaks of the connectionist approach as working with subsymbolic knowledge representations in contrast to the symbolic AI. Much effort has been expended to build better systems using artificial neural networks (ANNs). Unfortunately, much of this work has been performed using simulations of ANNs on digital computers (which themselves are symbolic processors). Any potential advantages that might come from using non-algorithmic ANNs are lost because of that.

The third view. Because none of the two approaches has achieved convincing results as to the essence of intelligence and awareness, a third view has emerged which stresses the fact that the brain is an example of a continuous nonlinear dynamic system (this point was already underlined by Turing in his 1950 article). Such systems cannot be simulated with arbitrary accuracy by digital computers. Accurate enough simulation is not even possible using analogue electronics. Proponents of this view contend that to achieve intelligence and even consciousness it is necessary to build a functional analog of the brain at the neuron level and to permit the needed complexity to emerge from a massive integration of many simple elements.

Another, related, reason for the perceived need for a third view is based on the Gödel's theorem (1931) about non-computable systems. Gödel has shown that in any sufficiently powerful logical system, statements can be formulated which can neither be proved nor disproved (are non-computable) within the system. Turing mentions this as an objection to the idea that machines could somehow think, but considers this irrelevant to his "imitation game." Nowadays, the opinion is widespread that, if not intelligence, than certainly consciousness belongs to the non-computable problems. For example, Penrose in "Shadows of the Mind" [1994] writes *"Appropriate physical action of the brain evokes awareness, but this physical action cannot even be properly simulated computationally"* (p. 12). Because of this, some researchers consider it even necessary to include quantum effects in the model that could explain consciousness [Hameroff & Penrose, 1996].[1]

3.1.4 What *are* AI?

We have seen that intelligence is a difficult to define concept and that AI is a diverse field. A unifying definition is difficult to give. Randall Davis in his article *"What Are Intelligence? And Why?"* [1998] suggests (after Aaron Sloman [1994]) conceiving AI as an exploration of a design space of intelligence. Davis sees this definition as useful because it emphasizes the multiple possibilities of what intelligence is (or are). Conceiving AI in terms of a design space suggests exploring and thinking about what kinds of intelligencies there are.

1. Turing mentions two other views concerning the question "Can machines think?". One is the theological view, which says that thinking is a function of man's immortal soul — Turing states that he is not very impressed by theological views. The other view is that the essence of thinking belongs to still undiscovered realms of physics (like extrasensory perception) — Turing finds this possibility plausible.

From the practical point of view, this approach may encourage development of applications that make use of many AI techniques simultaneously. In the following, three common AI techniques are presented, which exemplify the three views of what intelligence is: the logic/psychology view – the expert systems, the biology view – connectionist systems/neural networks, and psychology/biology view – case-based reasoning.

3.2 Expert systems

Expert systems clearly belong in the symbolic AI paradigm. The main premise on which the expert systems are based is that reasoning intelligently means reasoning logically. Logical reasoning is, in turn, usually understood to be a deduction using first-order logic (propositions, predicates and quantifiers). In accordance with this premise the initial research in this area concentrated on developing sophisticated inference systems, with emphasis on the reasoning and not on the knowledge.

These generalist programs, using wide but shallow knowledge, were not successful when applied to real world problems. Their strength lies with mathematical (algebra, logic) problems like the Newell and Simon's Logic Theorist which could prove theorems taken from Whitehead and Russell's *"Principia Mathematica."* More successful in dealing with something like real world problems were programs using deep knowledge of some narrow field like DENDRAL (1965–83, from Stanford University) for molecular structure analysis; MYCIN (1972–80, also from Stanford) for diagnosing certain infectious diseases; and Winograd's SHRDLU (1972) solving problems in a world of blocks (SHRDLU is also well known for its natural language capability).

While these programs were limited to a narrow application domain and are not "intelligent" in any broad human sense, they have proved to be useful in pragmatic and commercial terms.

Working of an expert system. Expert systems have two distinct parts: a knowledge base and an inference engine, see Figure 3.1a. The knowledge base contains knowledge about the problem domain, usually in the form of rules. The inference engine does the reasoning using the rules. The knowledge base is specific to the problem that has to be solved, while the inference engine is generic.

A typical inference engine operates on a set of facts (stored on a so-called scoreboard) using a set of rules [Giarrtano & Riley, 1989]. The inference cycle is shown schematically in Figure 3.1b. The cycle begins with instantiation and matching of the rule antecedents to the facts. All the rules

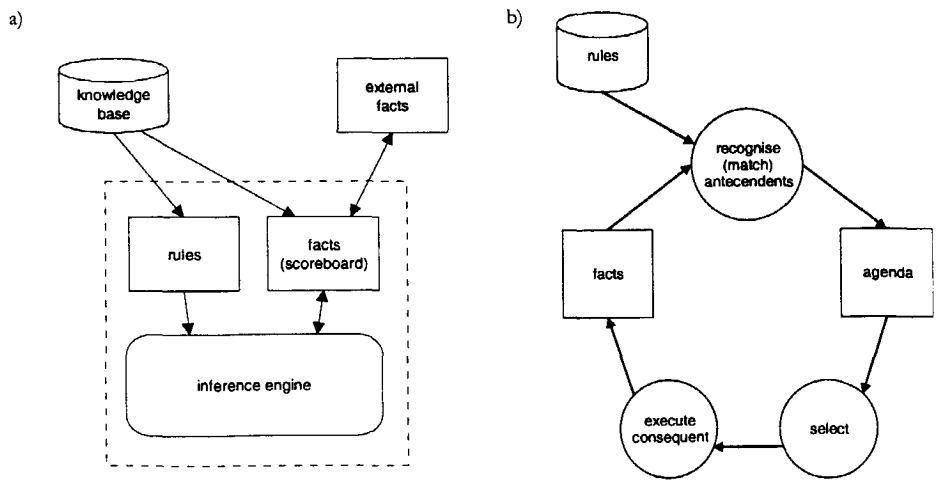


Figure 3.1: A typical expert system (a) and its inference cycle (b).

for which the antecedent part is true are placed on the agenda. From these rules the inference engine has to choose one rule whose consequent part will be executed. After executing the consequent new facts can be added to the scoreboard, facts can be altered or removed. Afterwards the reasoning cycle is repeated. If no new facts are present, or the solution has been reached, the reasoning stops. Both antecedents and consequents can be connected to procedures which carry out certain actions. These can be used for input, output, and processing of some external data to obtain new facts.

The above described inference cycle is referred to as forward-chaining. It is a deduction from the antecedents to the consequents of the rules. Another way of reasoning, called backward chaining, proceeds from the consequents to the antecedents and corresponds to hypothesise/test type of reasoning.

Representation formalisms. Apart from rules, other knowledge representation formalisms have been used in expert systems. The best known are semantic nets and frames. Semantic nets are used to represent propositional, declarative knowledge. The nodes in the semantic nets represent objects, concepts and situations. The edges represent relationships. Using the relationship information, it is possible to carry out inference on semantic nets. Semantic nets may be a well-suited tool for some problems as they provide both a convenient way of visualising the relationships and can easily be used for inference. However, their expressive power is limited, and as they get larger they start to pose problems both for representation and for computation.

A frame is a collection of named slots which can contain values, rules, procedures, other frames, etc. Frames provide a convenient method of representing typical objects (stereotypes) by assigning default values to some slots as well as by use of inheritance. Frames have a higher expressive power than semantic nets. Because the slots can hold rules they can also be considered more general than simple rule-based knowledge representations.

Advantages and disadvantages. As a tool for problem solving (rule-based) expert systems have a number of advantages:

- they represent an intuitive model of problem solving,
- domain knowledge is separated from "control knowledge" (inference),
- they are simple and self-documenting.

The more complex the system becomes the less clear some of these advantages are. For example, a rule-base of several hundreds of rules is hardly simple and self-documenting. What becomes especially problematic is the control of the dependencies between the rules and their sequence of firing. These and other disadvantages are listed below:

- "firing sequence" is opaque (selection from the agenda),
- rules can interact in surprising ways with each other (complexity),
- rules become complex the more *context* is taken into account,
- testing becomes difficult quickly with increasing size,
- difficulties in knowledge elicitation — low efficiency and reliability (real-world experts rarely think in terms of rules),
- difficulties in maintenance of the knowledge bases,
- may be ineffective in changing or unexpected situations, in the worst case making mistakes,
- may be unable to detect gaps in knowledge — ESs are usually built using the *closed-world assumption*[1].

1. When the closed-world assumption is used then everything is assumed to have some default set of properties, attributes, etc. unless otherwise stated, e.g., all objects may be assumed incapable of flying unless explicitly stated that they can fly [Cohen & Feigenbaum, 1982].

Expert systems in practice. Generally, expert systems are useful when: knowledge is well circumscribed, formalised, established and stable; common sense is not crucial; and there are acknowledged experts who find good solutions quickly while non-experts take very long or find no solutions at all. Expert systems can be expected to be most successful in situations where one already knows how to solve the problem (especially if the problem is routinely being solved) and the goal is to solve the problem faster and/or let anybody use the expert system to solve the problem (knowledge distribution). Expert systems solving simple, but often recurring, problems may be most successful (in economic terms).

It is necessary to able to describe a problem to be solved in a form suitable as input for an expert system. For example, one cannot just feed an expert system with images. Before an expert system can reason about images they have to be segmented and the image elements and their relations have to be described.

Expert systems are usually built using some commercial (or otherwise available) inference engine. Usually, such an engine comes with a shell that provides means for interaction with the rule-base. More sophisticated tools provide means for explanation, graphical rule editors, etc.

The largest expert system ever built is CYC® by Cycorp (www.cyc.com). Work on the system has started in 1984 and at present time, the knowledge base contains tens of thousands of terms with several dozen hand-entered assertions about/involving each term. The knowledge base contains facts about a broad range of topics, including, for example, definitions for Airplane, NuclearPoweredDevice, AbstractProgrammingLanguage, as well as definitions for Loneliness, Love, Affection.

3.3 Artificial neural networks

Artificial neural networks (ANNs) represent the connectionist AI paradigm. The driving force behind this paradigm was the idea of building information-processing systems similar to the ones found in biological organisms. An artificial neural network consists of a network of nodes (processing elements) connected via adjustable weights (connections). The weights can be adjusted so that the network learns some desired behaviour. The nodes perform simple operations like summing and passing a value through a transfer function. Usually, no interpretation can be given to individual nodes or connections (this is called subsymbolic representation). Such systems should have similar characteristics and advantages as the biological systems, namely:

- ability to learn from examples,
- ability to generalise,
- distributed-parallel processing,
- robustness (loss of individual nodes or connections does not significantly influence results),
- speed (in processing complex data, e.g., images).

ANN research began with networks implemented using analog electronics. Later, digital general-purpose computers were used to simulate neural networks. This meant sacrificing two of the advantages – robustness and speed. In fact, neural networks up to until recently have been known for notoriously long training times. Though, over the years, there were various implementations of neural networks using special hardware – usually parallel digital processors with a special instruction set – most of the neural network implementations are for normal SISD digital computers. Various improvements of neural network algorithms (aimed usually to increase speed) moved them further and further away from their biological protoplast. This puts them often in the same categories as non-AI techniques used, for example, for function fitting, data classification, or clustering (see Section 3.6.2).

In the following sections several examples of typical neural network types are presented. They are distinguished based on the way they are trained — supervised and unsupervised trained. More detailed description of all the architectures presented here, except the radial basis function network, can be found in [Zurada, 1992].

3.3.1 Supervised trained networks

In supervised training the network is presented with input-output pairs and has to learn the (functional) relation between the input and output space.

Perceptron. Perceptron is a simplest type of neural network and was introduced by Frank Rosenblat in 1959. A schema of the perceptron is shown in Figure 3.2a. It consists of a layer of fixed input nodes which can do simple processing of the data (e.g., computing a majority function), an output node which sums its input and applies a threshold function to it, and a set of variable weight connections between the input nodes and the output node. Perceptron can be used to recognise patterns on the input matrix (e.g., an image). The correct values of the connection weights are found using the delta (Widrow–Hoff) rule — it is called so because in every learning iteration a certain portion of the output error is removed.

A disadvantage of the perceptron is that it cannot solve non-linearly separable problems (e.g., XOR). This holds at least for the diameter-limited perceptron, it is always possible to construct input units making solution of specific problems possible [Aleksander & Morton, 1991].

Multiple-Layer Perceptron (MLP). MLP (in its various forms) is the most often used neural network type. It is similar to the original perceptron but has an extra hidden layer (Figure 3.2b). Because of that, it can solve non-linearly separable problems. In fact, if continuous transfer functions are used then it can approximate any continuous function $\Re^n \rightarrow \Re^m$, where n and m are the numbers of input and output nodes; or construct arbitrary nonlinear decision boundaries (provided an adequate number of hidden nodes are used) [Cybenko, 1989].

The simple delta rule cannot be used to train the MLP because the errors on the hidden layers are not known. In 1986 Rumelhart, Hinton and Williams suggested a method to calculate the error of the hidden layer [Rumelhart et al., 1986]. In principle, this requires the use of a differentiable transfer function so that the error can be backpropagated using the chain rule for differentiation. Most training methods use error backpropagation to determine the gradient of error with respect to weight change.

Training methods based on the error gradient are usually simple gradient descent (generalised delta rule). They are simple in implementation, but require many training iterations. The training can be speeded up using some heuristic improvements like a momentum term or a variable learning

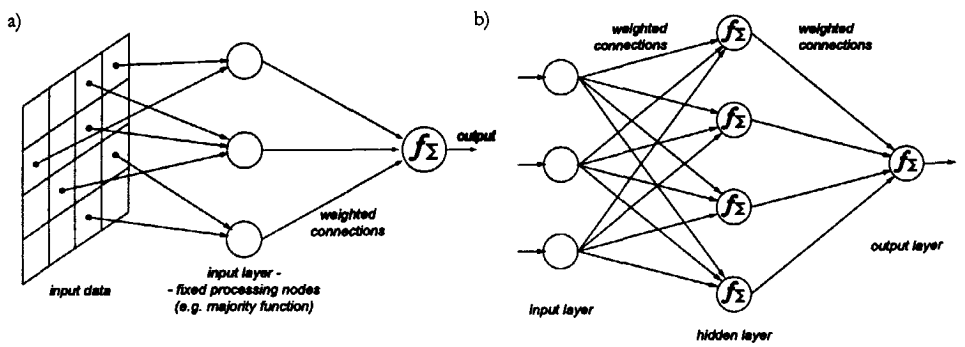


Figure 3.2: Perceptron (a) and multiple layer perceptron (MLP) (b).

rate [Jacobs, 1988]. Quickprop is a variation of the standard gradient-descent learning. At each iteration, instead of taking a step proportional to the error gradient, a step to the minimum of a local error-approximating parabola is taken.

The conjugate-gradient method is an improvement over the simple steepest descent. It is also a first-order method, and its storage requirements are of the same order as in case of steepest descent. More calculation is required per training iteration, and the speed gain is only significant when low training errors have to be achieved. Many second-order methods have been investigated for training MLPs, like: Davidson-Fletcher-Powell (DFP), Broyden-Fletcher-Goldfarb-Shanno (BFGS), and Levenberg-Marquardt minimisation [Watrous, 1987; Bauman, 1994]. They are significantly faster than the first order methods; however, they are second order in the space requirements, which may limit their usefulness to training small networks only. Also, they are far away from the initial idea of a neural network, i.e., distributed processing, and in principle are just a way of function fitting. Still other training methods may use random optimisation techniques (sometimes called chemotaxis) [Baba, 1989; Montague et al., 1992].

In general, MLPs are good at interpolating data, but may give large errors on extrapolated data. In classification MLPs are popular because they can learn complex decision surfaces; however, they have also disadvantages, like a large number of parameters and a long training time for the most commonly used backpropagation training algorithm. In these respect radial-basis function networks (RBFN), described next, may be a better choice as they are faster to train and many of the configuration parameters can be determined automatically.

Radial Basis Function Network (RBFN). RBFNs have an architecture similar to neural networks; however, the algorithms they use are significantly different from those used in other neural networks and it would be difficult to find biological basis for them.

A RBFN (see Figure 3.3) consists of a nonprocessing input layer, a layer of non-linear processing nodes (usually containing a multivariable, Gaussian-like function) and an output node performing linear summation. The main parameter defining a radial basis function is centre c_i. The other parameters are: the type (like a Gaussian or a logistics function) and the size (for instance as defined by the covariance of the Gaussian function). RBFNs, in their simplest form, are trained using the linear least squares (LLS) algorithm to find the values of the weights w_i. If also parameters of the basis functions have to be adjusted then generalised linear least squares (GLLS) and/or gradient descent algorithms have to be used.

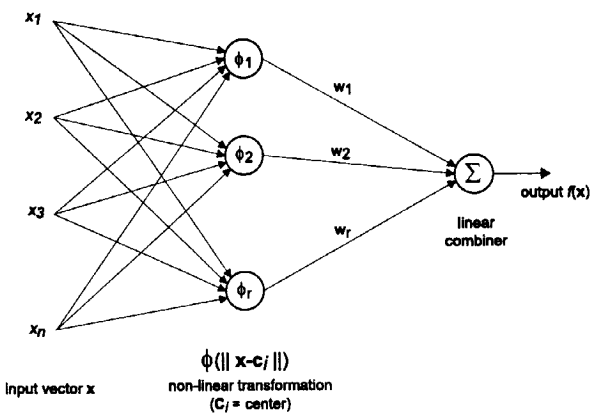


Figure 3.3: Radial basis function network

An advantage of the RBFN is the short training time, for example, Nair et al. [1993] report that while it took 4 hours to train an MLP (using backpropagation), a comparable RBFN could be trained in only 2 minutes. Additionally, RBFN training lends itself to a better mathematical analysis than MLP training. Training algorithms can be devised, which use a combination of well-understood techniques, like clustering and least-squares fitting. Training can also be fully automated, i.e., freeing the user from the need to choose the best number of nodes, the learning rate, etc. An example of such an automatic training algorithm, which automatically chooses the number of functional nodes (centres), can be found in [Chen et al., 1991]. Interesting ideas on the RBFN training, for example choosing centres based on clustering of input data and adjusting parameters of radial basis function by gradient descent (in addition to least-squares fitting), can be found in [Bengio & De Mori, 1991].

Generally, RBFNs can be used for the same purposes as MLPs and are in many applications replacing MLPs. Still, MLP is a more general and theoretically more powerful network type.

Hopfield network. The Hopfield network is a recurrent, (almost) fully-connected, autoassociative network, see Figure 3.4a. Being autoassociative means that the network is trained using vectors which simultaneously are the input and the desired output. In the Hopfield network, the input and output consist of binary patterns. The training consists of computing the weights of the nodes based on the values of the patterns the network has to recognise. The execution is asynchronous. The original Hopfield network, after having been trained on a prototype set, is able to recall the prototype which is most similar to the currently presented input.

Hopfield network seems, at first, to be an interesting candidate for recognition of similar images; however, it requires full interconnection of nodes what makes implementations for large images unpractical. And, what is probably more important, the similarity measure implemented by the Hopfield network is the Hamming distance which is too simple for real world images [Wang, 1993].

Working of the Hopfield network can be analysed in terms of minimising energy, where the energy minima are formed by the training prototypes. One disadvantage of the Hopfield network is an existence of local minima other that these corresponding to the prototypes, which means that for some inputs the network may not converge to one of the desired end states. A modified Hopfield network which overcomes this problem is called a Boltzman machine. While in the Hopfield network there is a hard threshold which precisely determines if a given neuron fires, in the Boltzman machine the neurons have a certain probability of firing. This probability is defined by a function which depends on the activation value and on the so-called temperature parameter. The higher the temperature the more random the firings are. A zero temperature corresponds to the Hopfield

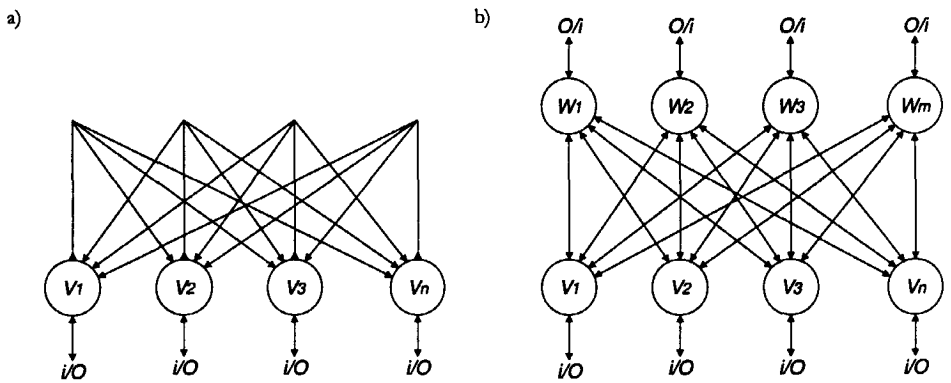


Figure 3.4: Hopfield a) and BAM b) networks. In Hopfield network the same nodes are used for input and output. In BAM nodes can serve exchangeably as either input or output. Connection weights assume −1 or +1 values.

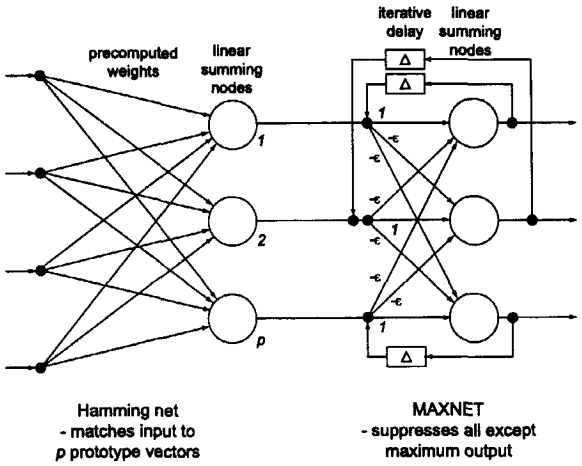


Figure 3.5: Hamming net and MAXNET.

network. When presented with an input pattern the temperature is high and it is slowly lowered as the network converges. The randomness allows the network to exit the local energy minima. This way of finding a minimum is called simulated annealing.

The energy minimisation model used in the Hopfield network can be used to solve a variety of minimisation problems other than the original idea of the most similar pattern determination. Two examples of the use of the Hopfield net for this type of problems are: the well-known Travelling Salesman problem, and fitting of multiple lines through a collection of points presented in [Kamgar-Parsi & Kamgar-Parsi, 1990].

Bidirectional Associative Memory (BAM). BAM network (Figure 3.4b) is similar in structure to the Hopfield network, only there are two sets of nodes — one set for input and one for output. The network is trained on pairs of patterns (thus it is a heteroassociative net) in similar way as the Hopfield network. When presented with an input similar to one element of some pattern pair, it is able to reconstruct the whole pair. It can operate in either asynchronous or synchronous mode. Usually, one set of i/o nodes is used for the patterns to be recognised and another for the class codes. BAM requires fewer connections than the Hopfield network to store the same number o pattern pairs[1], but it is also less noise tolerant.

Hamming net and MAXNET. A combination of Hamming and MAXNET, see Figure 3.5, can be used for classification. A Hamming net calculates the Hamming distance between an input vector and a set of prototype vectors. MAXNET is a recurrent network which given some input vector extinguishes all values except the maximum one. If both networks are combined, a minimum-Hamming-distance classifier is obtained.

3.3.2 Unsupervised trained networks

Unsupervised trained networks are trained only on the input patterns and they can themselves discover patterns and/or clusters in data.

1. The upper bound for number patterns that can be stored in a Hopfield net is given as about *0.1n*. The upper bound for BAM is given as *min(m,n)* (or more conservatively as *sqrt(min(m,n))*).

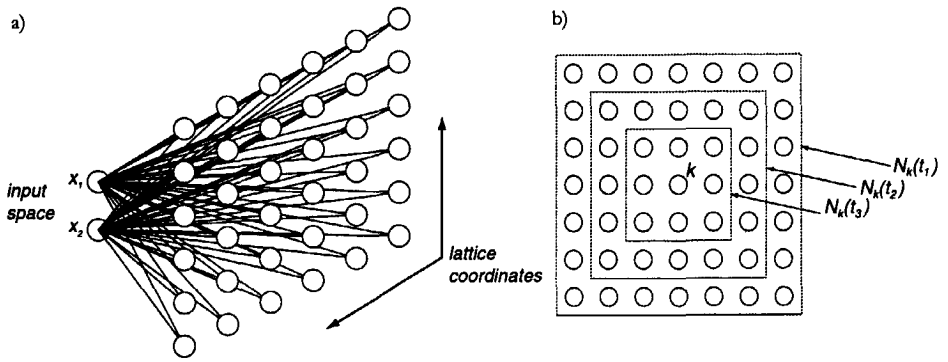


Figure 3.6: Kohonen self-organizing map. Mapping of the input space in to a lattice of nodes (a). Definition of neighbourhood in the lattice (b). Neighbourhood radius depends on time *t*.

Kohonen network (vector quantisation). Kohonen network (not to be confused with the Kohonen feature map described further) classifies input vectors into a specified number of K categories according to the clusters detected in the training set. The measure of similarity used is a normalised scalar product. This network represents winner-take-all learning. It can properly detect only linearly separable clusters.

Cluster discovery network (ART1). In the ART1 network architecture two stages can be distinguished: the forward and the backward. The forward stage is similar to that of the Hamming and MAXNET, only the similarity and the maximum determination are done by the same nodes. The backward stage consists of connections used to check the similarity of the winning pattern to the input pattern — if the difference is too large, a new node corresponding to the new pattern is added to the network.

Thus, ART1 can accommodate new clusters without affecting the clusters already learned. While ART1 operates on binary data, the improvements, ART2 and ART3, can work with continuously valued input vectors [Wang, 1993]. ART1 type of network may produce degenerate results when trained on noisy data because spurious new clusters may be formed [Sarle, 1994].

Self-organizing feature map (SOM) by Kohonen [1989]. The Kohonen map maps an n-dimensional input space into a discrete lattice of formal nodes. Mapping is done by connecting every element of the input vector to every node in the lattice via adjustable weights $\boldsymbol{w}$, see Figure 3.6a. The nodes perform matching of the input $\boldsymbol{x}$ and the weight $\boldsymbol{w}$ vectors. For the lattice, a topological neighbourhood applicable to all the nodes is defined, like in Figure 3.6b. The SOM is trained to output localised responses, that is, an n-dimensional input value results in a local activity (higher output values of the nodes) in the output lattice. Training is done in a number of iterations until a stop criterion is met. During each iteration, first, the input is applied to the map and the best matching node (one with the highest output) is determined. Then, the weights in the neighbourhood of that node are adjusted using a learning rule. The learning constant in the rule depends on the distance from the centre of the neighbourhood (it decreases with the increasing distance) and on the time from the beginning of training (it decreases as the training progresses).

Once trained, the map will respond selectively to different output; however, in such a way that similar input will activate neighbouring areas of the map. SOMs are useful for analysing multidimensional data sets in order to discover patterns in data. For example, Kohonen used it for phoneme signal classification, and Grimaldi [1995] reports use of SOMs for analysis of eddy-current data.

3.4 Case-based reasoning

Case-based reasoning (CBR) systems will be described in more detail in Chapter 4, this section contains only a short sketch of the most important characteristics of CBR. Because CBR makes use of many AI and non-AI techniques it is often called a methodology and not just a technique [Watson, 1998].

Case-based reasoning is relatively new in AI. Origins of CBR date to the end of the 1970s, with more widespread research beginning in the second half of the 1980s. Case-based reasoning is based on an observation that people usually solve new problems by reusing (old) solutions to similar problems and that the knowledge they have is rarely organised into neat rules but, instead, in whole groups of related events, situations, actions, images, etc. [Riesbeck & Schank, 1989].

CBR systems base their solutions on previously solved problems (cases) which are stored in a case-base [Watson & Marir, 1994]. When a new problem is presented to a CBR system a similar case (or cases) is (are) retrieved from the case-base. Depending on the differences between the retrieved and the presented problem, the retrieved solution may have to be adapted to obtain a solution to the new problem. The solved problem may be retained in the case-base if deemed useful.

An attractive facet of a CBR system is its inherent ability to learn. This means that a system can be put into operation with a minimal set of solved cases in the case-base. The case-base is filled with representative cases as it is used. CBR is also a methodology which may offer a solution to the problem of knowledge elicitation and maintenance. The knowledge elicitation is simplified because often it can be reduced to collecting the cases. The maintenance is easier because, on the one hand, CBR systems can adapt to changes, and, on the other hand, the knowledge contained in the cases is easier to comprehend.

Most of the CBR systems operate in a close cooperation with a user. Sometimes this is a design choice dictated by the intended application domain, and sometimes it is a necessity because of missing or very simple adaptation stage present in the system, which in turn may be a result of problems in formalising adaptation knowledge.

3.5 Other AI techniques

3.5.1 Reasoning with uncertainty

In real-world problem solving one has to deal with uncertainty and ambiguity. Many techniques have been developed for reasoning with uncertainty, like: certainty factors, Dempster-Shafer theory, fuzzy logic, Bayesian networks. (A description of all these techniques can be found in [Giarratano & Riley, 1989].)

Certainty factors and Dempster-Shafer theory. One method of reasoning with uncertainty is using certainty factors (CF). This technique was used for the first time in MYCIN, an expert system for medical diagnosis. A certainty factor (–1...+1 value range) of some hypothesis due to some evidence is the difference of the measure (0...1) of the increased belief in the hypothesis and the measure (0...1) of the increased disbelief in the hypothesis due to that evidence. A positive CF means that the evidence supports the hypothesis, a negative CF means that the evidence favours negation of the hypothesis, a CF equal zero mean that the evidence is inconclusive. Measures of belief and disbelief are important mainly during knowledge acquisition for assigning CF to the rules. During working of the system only single CFs are used. A number of formulas how to calculate CFs are defined for AND, OR, NO, and IF...THEN... operators, as well as for combining CFs from several rules concluding the same hypothesis. Knowing the certainty of the initial data, and of course the certainty of the rules, it is possible to calculate the certainty of the final result of the reasoning.

While certainty factors represent a rather ad hoc approach to inexact reasoning the Dempster-Shafer theory has a good theoretical foundation. Dempster-Shafer theory assumes that there is a fixed set (a frame of discernment) of mutually exclusive and exhaustive elements (answers, conclusions) which are the subject of the reasoning. Also, there is a certain fixed (=1) amount of belief that is distributed over all the subsets (including the empty subset – the environment) of the frame of discernment. New evidence usually causes *redistribution* of the belief among the subsets. The way it is done is defined by the Dempster's Rule of Combination. Sometimes during combination the sum of belief falls below 1, it has then to be normalised so that the sum is always 1. The evidential interval of a given subset S is defined as [*Bel(S)* ... *1-Bel(S')*] where *Bel(S)*, the belief measure, is the total belief of a set S and all its subsets, and S' is a complement of S. The evidential interval can be expressed as [evidence for support ... evidence for support + ignorance] or in other words as an interval between pessimist and optimist value of the belief, the true value assumed to be somewhere in-between.

Fuzzy logic. The weak points of the Boolean first-order predicate logic are vaguely defined propositions which are neither fully true nor fully false. Examples of these type of propositions are: *a curve is fat, a signal has sharp peaks, a signal is noisy*, etc. Such propositions are a common part of human reasoning. In 1965 Lofti Zadeh has published an article [Zadeh, 1965] about fuzzy sets that provided the foundation for the fuzzy logic which is capable of dealing with vague propositions. A normal (crisp) set contains only elements which exactly satisfy the set properties, these properties in turn can be expressed as a membership function mapping the domain into 0 or 1. A fuzzy set allows for a partial element membership; thus, the membership function has a 0...1 range. For example, one could define a set of "fat-looking curves" where the membership value would depend on the ratio of the sides of the smallest enclosing rectangle. Fuzzy propositions may also have fuzzy quantifiers like *most, very, usually*, etc. These usually are interpreted as modifying the membership functions of the fuzzy sets to which they apply.

Just as classical logic forms the basis of conventional expert systems, fuzzy logic forms the basis of fuzzy expert systems. Logical operators for multivalued logic as well as ways of interpreting fuzzy rules (fuzzy modus-ponens) can be defined. Both for the logical operators and for the modus-ponens many possibilities exist, see e.g. [Bouchon-Meunier, 1991].

Bayesian belief networks. Bayesian belief networks (BBN), called also simply belief networks or causal probabilistic networks, are used to model domains containing uncertainty. BBNs give structure to probabilistic models, which have exponential complexity, and make them more tractable [Koller, 1998]. A BBN is a directed acyclic graph where each node represents a random variable. The links represent relations of dependence between the variables, and the paths between the variables let us draw conclusions about independence of the variables (knowledge of the independence is important as it simplifies reasoning using a given Bayesian network). Each node contains the states of the random variable it represents and a conditional probability table (or in more general terms a conditional probability function). The conditional probability table contains probabilities of the node being in a specific state given the states of its parents. Bayesian networks can be used for hypothetical reasoning as to probability of, for instance, a certain hypothesis H_i being true given some evidence E — the inference is done using the Bayes theorem:.

$$P(H_i|E) = \frac{P(E|H_i)P(H_i)}{P(E)}$$

No single technique for reasoning with uncertainty can be deemed as the best one. Every one of them has some deficiencies:

- reasoning based on probability (Bayesian networks), though theoretically correct, may be against our intuition,

- certainty factors are considered an ad hoc measure, and may lead to incorrect results for deep reasoning chains,
- Dempster-Shafer theory of evidence requires extensive computation and the results of its normalisation procedure may sometimes be counter intuitive,
- fuzzy reasoning is subject to many arbitrary choices, e.g., concerning implication and modus-ponens operators.

However, the general conclusion is that when the depth of reasoning chain is limited to a few levels only then the choice of the technique for reasoning with uncertainty is not critical. When expert knowledge has to be captured it may be advantageous to choose fuzzy reasoning because it also offers a comprehensive method to handle uncertain and imprecise data as well as linguistic terms as used by the expert(s). When knowledge is in the form of data it may be used to train a BBN. Techniques for reasoning with uncertainty can often be combined with the techniques described in the previous three sections (ANN, ES, and CBR).

3.5.2 Machine learning

Learning can be defined as improving performance at some task based on experience [Mitchell, 1997]. Another goal of learning can be "concept formation", which may lead to improved performance, but may also be a goal in itself, e.g. in data analysis. Though, generally, learning is considered as one of the aspects of intelligence, not all AI techniques are capable of learning, the obvious example being expert systems (at least in their common form). The best known machine learning (ML) techniques are described below.

Artificial neural networks. The ANNs learn the functional mapping or they discover relations within the data from a set of examples they are presented with (see Section 3.3).

Decision trees. A decision tree is a classifier (or a discrete value functional mapping) which, given some input in a form of a set of attribute-value pairs, determines the output value using a series of decisions based on the values of the attributes. The order of the attribute value tests is fixed in a form of a tree (see, for example, Figure 7.18). There are several algorithms available which can be used to construct (induce) a tree based on some data set. Typical examples are the ID3 and C4.5 algorithms developed by Quinlan [1993]. These are greedy search algorithms that construct the tree recursively, at each step choosing the attribute which is to be tested such that the separation of the data examples is optimal. The major differences between training algorithms lie in the measure used to determine the most optimal attribute to be tested, the algorithms can use, for instance, measures based on information content or reduction of entropy. Variations of the simple decision tree have been developed which can use a linear combination of the attributes in the decision criterion, or which are capable of continuous functional mapping (regression trees).

If-Then rules learning. Propositional (variable-free) rules for a rule-based classifier can be learned by rule induction from a set of examples. Essentially, the same algorithms are used as in the case of decision trees (it is always possible to represent a decision tree in the form of a set of rules).

Inductive logic programming (ILP). ILP is a method of learning first-order rules, i.e., rules that may contain variables. Given a set of predicates about the data (ground literals), a set of known functions and rules (Horn clauses[1]), an ILP algorithm can induce a theory (a set of Horn clauses) which is a generalisation of the relations discovered in the data set. The power of ILP lies in the ability to produce efficient generalisations and the use of a-priori knowledge.

1. ILP systems are usually implemented in PROLOG.

Instance-based learning. Instance based methods do not construct general descriptions of the knowledge, but simply store the training examples. Generalising, over a (small) subset of stored examples, is postponed until a new data instance must be classified (that is why they are also called "lazy learning" methods, see Section 4.6). Given a new instance, its relationship to the already stored examples is examined in order to assign the classification. Examples of instance-based learning are: *k*-nearest neighbour learning, locally weighted regression, and CBR. Instance-based learning methods are disadvantaged by their computation and memory requirements. A major advantage of the instance-based learning is its reliability; because they store the whole training set they can recognise when they have to generalise beyond the training data.

CBR, even if it does not have an adaptation stage, distinguishes itself from the other instance-based learning methods by the use of more complex representation forms so that individual instances can represent complex problems.

Genetic algorithms. Genetic algorithms (GA) are an optimisation technique. One thing that distinguishes them from other optimisation algorithms is that GA work on a large set (a population) of possible solutions simultaneously. At each iteration of the algorithm the best solutions in the population are selected. From this selection the next generation of the solutions is produced by combining and exchanging parts of the solution between the members of the selection. Random changes to the solution (so-called mutations) are also allowed. The main application of GA is in search in complex problem spaces where the solution is composed of several parts, so that it makes sense to concentrate search on certain hyperplanes of the search space once parts of the optimal solution already have been found. A typical example of such a problem is route optimisation.

Reinforcement learning. Reinforcement learning is often used for control problems. Given some unknown system which is to be controlled and some initial state, a control action or a sequence of actions has to be found such that another, more optimal, system state is achieved. Because the system is unknown, the appropriate control actions are also unknown (for a known system an inverse of the system could be used to determine the control action). However, the optimality of the state is generally easy to determine. In reinforcement learning, a control agent is being trained to find the optimal action, given some current and desired state, only by telling it if an action has lead to a good state (agent receives a reward) or if an action lead has to a worse state (penalty). This way the positive actions are reinforced and the agent improves its performance (learns). Reinforcement learning is difficult because often a long delay is present between the actions taken and the evaluation of the result (e.g., in game playing).

Overall, use of learning systems has two main advantages. First, the construction cost may be lower[1], certainly if the system is not constructed from scratch but uses existing software tools and components. Second, a learning system is able to adapt to changing environment. However, most of the algorithms require a special training phase which may make on-line adaptation ("sustained learning" [Aamodt & Althoff, 1993]) difficult. A technique which is well suitable for on-line learning is instance-based learning.

A disadvantage of many ML techniques is the fact that they are data oriented, i.e., they model the relationships contained in the data set. The resulting model may sometimes differ from the actual domain theory, usually, when the data set is not a representative selection from the problem domain. Only few ML techniques allow for use of a-priori knowledge. ML techniques also have problems with noise. Though, many of them have special provisions to prevent them from noise

1. Turing [1950] considered using learning (teaching a "child machine") as an alternative to "manual" programming of intelligent systems. According to his (optimistic) estimate it would take 3,000 man-years to program a computer capable of passing the Turing test. It should be remembered that in 1950 computers were being programmed in machine code — 0s and 1s.

fitting, these may have a side effect of ignoring rarely occurring but possibly important features of the problem domain.

3.6 Choosing the right technique

3.6.1 Exploring a space of techniques

As already mentioned in Section 3.1.4. it is advantageous to think about solving a problem as searching in a space of techniques which provide means of solving that problem. In this section we are mainly concerned in the space of AI techniques, but a sound approach would require including useful non-AI techniques in the possibilities considered, this is discussed in the next section.

In this chapter a number of AI techniques have been presented from which it is possible to choose when making an "intelligent" system. Of course, one is not limited to a choice of just one technique, as often use of a number of different techniques, each solving a certain subproblem, brings the best results (see Section 5.4.5). Evaluation of the suitability of a technique can be done based on a number of criteria, some of which are:

- *Type of the problem to be solved* — Is it a planning, a classification, or a diagnosis problem? Some techniques are better suited for some problems then for others. An MLP is certainly not the best choice for planning, but a Boltzman machine could be used to solve some planning problems.
- *Form of available knowledge* — Are there experts, rules and procedures, data records, models available which describe either the problem, the solutions, or ways of solving the problem? It is necessary to determine if this knowledge is really or only theoretically available. For example, there may be experts present but knowledge acquisition maybe difficult or expensive because it detracts the experts from their normal work (see the following point about cost); data may be available but it is not well documented. Models of the problem domain are usually the most valuable because if a good model is available, a system can be built that should be able to solve problems quite different from the ones seen so far. However, some models, especially of physical phenomena, can be expensive in computation. If we limit the choice to the three techniques discussed so far, than if the knowledge is in the form of data records then ANNs might be an appropriate technique; knowledge in form of expertise or causal models suggests use of rule-based expert systems; knowledge being a combination of data, models, and expertise could be best utilised by CBR systems.
- *Acceptable cost of constructing the system* — One has to determine if a system fulfilling all the requirements can be constructed within the budget. A significant, and often underestimated, portion of the cost lies in the knowledge acquisition. Cost can be significantly reduced if a use is made of familiar techniques and of available software (own or third-party). Often, so-called 80:20 solutions (e.g., "intelligent" assistance of the operators instead of replacement) provide satisfactory solutions at acceptable cost. Also, the cost of maintenance should not be underestimated.
- *Automatic operation vs operator support* — Is the system to be fully automatic/autonomous or will the system mainly support a human in decision taking? On the one hand, designing the system for the assistance may make the design simpler because, in principle, the system can rely on the user for solving problems it cannot solve itself. It also opens the possibility of learning from the decisions made by the user. On the other hand, one has to pay more attention to designing a good user interface, as it may ultimately decide about the success or the failure of the system.
- *Necessary reliability* — Is one satisfied with 90% correct solutions, or are any failures critical? If reliability is not crucial, one has a much wider choice of the techniques. If it is crucial then techniques which make wide generalisations (e.g., ANN) may be unsuitable. In general, CBR is a reliable technique.

- *Maintenance* — Will the maintenance of the system be necessary? Will the maintenance be done by the end-user or by the system developer (this implies additional costs)? If it is known that the system will require maintenance it has to be designed for it. For example, the end-user cannot be expected to edit rules in an expert system if the parameters of the problem change.
- *Required speed* — Will the application be working real-time on-line? Will it be working interactively with the user, or is it enough that the processing is done in batch mode? Achievable speed is significantly influenced by the technique used, for example, ANN classifiers are much faster than instance-based classifiers. Speed is especially critical for interactive system, as then the worst case speed is decisive; for systems doing batch processing only the average speed is important. Some speed-up can be achieved by using faster hardware, but generally better algorithms provide larger speed gain.
- *Uncertainty* — Will the system have to deal with uncertain or noisy input data? Is the knowledge ambiguous? In such cases one of the techniques for reasoning with uncertainty may have to be used. Also systems using ANNs may cope well with uncertainty by choosing proper output ranges for the networks and, for instance, training them to output values of membership functions. CBR is also be able to handle uncertainty, if it is taken into account when designing a system, for example, by choosing an appropriate similarity measure for the retrieval.

3.6.2 Maybe it does not have to be AI?

Many ANN types, some ML algorithms, and some types of retrieval algorithms in CBR have close relation to techniques used in the statistics. When building a system it is good to be aware of them because statistics offers a set of useful tools, for instance, for the analysis of the validity of some assumptions, fast parameter fitting algorithms, etc. Also, statisticians realise the assumptions about the data they use, unfortunately, users of neural networks often do not. In fact, still, the assumption is popular that "neural networks are intelligent they will figure it out" no matter what data they are given.

ANNs are often used for classification, a task for which many statistical classifiers have been developed. For certain problems other than ANN classifiers may be more appropriate. Examples of typical classifiers and their comparison can be found in Section 5.3.4. Sarle [1994] gives a number of correspondences between various ANN types and statistical techniques, these are listed in Table 3.1.

Table 3.1: Some examples of the equivalence between ANN and statistical techniques.

ANN	statistical techniques
perceptron with linear transfer function	linear regression – multivariate if more than one input node is present, and multiple if more than one output node is present
perceptron with sigmoid transfer function	logistics regression
Adaline (threshold transfer function)	linear discriminant function
multilayer perceptron (MLP)	nonlinear regression
Kohonen network (winner-take-all)	k-means clustering
radial basis function network (RBFN)	kernel regression
ART1	several iterative clustering algorithms

3.7 Building AI applications

AI systems are not different from other software and they will be successful only if there is a real demand for them. In case of AI this is especially important because AI applications usually have to be developed with cooperation of users, because of the need for knowledge acquisition. If the users are not convinced of the need for the system, they will not cooperate.

Because AI techniques are often computation intensive, early in a project the hardware constraints (the deployment platform) should be determined. Sometimes the chosen hardware may influence the choice of the technique, and sometimes the chosen technique will determine the hardware requirements. One should remember that during long projects the hardware and the software tools available change.

It is usually wrong to definitively choose the technique at the very beginning of the development project, even before detailed problem analysis has been done. As obvious as this might seem, this rule is often not followed, and frequently one tries to force some AI technique (usually, the one the developer is familiar with) on a problem. However, users are not impressed with the technology used. In fact, the simpler the system, the easier to understand it is and, as a result, it is also easier to accept.

Because AI systems are supposed to be "intelligent" the users have high expectations about their accuracy, and the systems should live up to these expectations. Many AI systems operate automatically processing huge amounts of data. Users should be able to depend on these systems, because constant supervision is unacceptable.

A good discussion of all these aspects based on a successful AI project – a diagnosis system for tank engines — can be found in [Helfman et al., 1998]. Examples of several failed AI projects are discussed in [Moulton, 1998].

3.7.1 Knowledge acquisition

After the type of knowledge available has been determined, and the choice of the AI technique has been made, usually, the next step in building an AI system is knowledge acquisition. Though this term is usually used in the context of building expert systems, it is equally applicable to other AI system like ANNs and CBR.

It is important that the AI developer (knowledge engineer) should become familiar with the problem domain. At the beginning this is usually limited to the terminology used, but at the end of a project the developer usually becomes a sort of expert himself. Also, ideally, someone who has sufficient expertise in the problem domain should be assigned to the AI project, so that his time is always available, certainly if the problem to be solved is complex.

If the chosen technique is ANN then the data is everything. It should contain the information relevant and necessary to solve the problem. There has to be enough data present to be able to make any useful judgements about reliability of the system. The type and the amount of data determines then the ANN configuration. Often, preprocessing (e.g., feature extraction) is necessary to transform the data in a form suitable for ANN input. Sometimes, finding the right preprocessing (feature extraction) method constitutes the crux in solving the problem.

In case of expert systems the situation is optimal if there are existing rules and procedures, for instance, for diagnosis, classification — these can often be directly translated into rules. CBR systems can be implemented quickly if good records of solved problems are available (more about knowledge acquisition for CBR in Section 4.11.3).

It is a good idea to perform the knowledge acquisition iteratively. A simple working prototype should be built as early as possible so that an expert can "play" with the knowledge already contained in it. This way any misconceptions, gaps, contradictions, etc. in the knowledge become early apparent. Having a working piece of software the developer has something to talk about with the expert and can ask more precise questions. The encountered problems may remind the expert

about things he had forgotten to mention in previous knowledge acquisition sessions or things he thought to be self-evident, etc. The feedback from the user serves then to improve the prototype, which then again can be tested by the expert.

3.7.2 Implementation

Reuse. New AI systems could be built faster and cheaper by making use of existing AI systems – *including* the knowledge they contain. One could say that much of the progress in practical uses of the AI has been hampered by building AI systems from scratch. This may be partly caused by the fact that AI systems are not that intelligent after all. They miss flexibility — that part of the intelligence that allows humans to recognise similarities between problems and find solutions for problems they have never seen before. AI systems are often specifically tuned to their specific application areas to achieve optimal results at minimal cost (flexibility would often mean going beyond what the specs require).

A recent development that might increase the amount of component reuse is the emergence of the so-called intelligent agents [Maes, 1994] — small programs capable of autonomously solving small problems in a varied environment. Provided that they are equipped with proper (standardised) ways of exchanging information one can imagine them cooperating on solving more complex problems.

User Interface. For systems which work interactively with the user the appropriate user interface is often critical to the success of the application [van de Kraats, 1989]. If an AI system is to support a user, for example, in data interpretation or a design task then it is important that the suggestions are clearly presented and that the corrections can easily be made by the user. Thus, UI should be an integral part of the system design. This is especially obvious in the case of CBR systems but also expert systems profit from easy-to-use explanation or exploration facilities.

In the case of some interactive learning systems the system will generally learn better if the user provides more information about the proper solution and reason for failure. This will improve the system in the long time, however, will require much work from the user at the given moment. The question is: How much interaction the user really wants? Usually, the user wants to be helped and not to spend time helping the system to learn (unless the advantages are significant and immediately visible).

As already mentioned, because AI systems are supposed to be intelligent, experienced users tend to get annoyed if the questions asked by the systems seem too stupid, too detailed, etc. On the other hand AI systems are supposed to be used by non-experts to solve complex problems; therefore, AI systems should be able to make it clear to the novice users what information they require. Thus, it makes sense to implement the UI so that it can accommodate the level of expertise of the user.

3.7.3 Maintenance

An important aspect of AI systems is maintenance. It includes correcting faults that could have occurred during the knowledge acquisition and were not detected during prerelease testing, and adapting the system to changing conditions. Learning systems though capable of adjusting to changing conditions also require maintenance. Often, the learning does not occur automatically but has to be initiated by the user. Moreover, the knowledge acquired by the system has to be verified.

The system maintenance becomes really difficult if it has to be done by the end-users, however, it is feasible if the system is equipped with proper tools. Thus the issue of maintenance should be considered early in the design process, and not have to devise ad hoc solutions after the system has been deployed.

3.8 References

Aamodt, A. and Althoff, K.-D. (1993) "Problem Solving and Sustained Learning from Experience: Analyzing Methods with Respect to Domain Characteristics", M. van Someren (ed.), *Proc. ML-net Workshop on "Learning and Problem Solving"*, Blanes (ftp://ftp.ifi.ntnu.no/publikasjoner/vitenskaplige-artikler/mlnet93.ps.gz).

Aleksander, I. and Morton, H. (1991) *An Introduction to Neural Computing*, Chapman & Hall.

Baba, N. (1989) "A new approach for finding the global minimum of error function of neural networks", *Neural Networks*, Vol. 2, No. 5, pp. 367–373.

Bengio, Y. and De Mori, R. (1991) "Connectionist models and their application to automatic speech recognition", *Artificial Neural Networks and Statistical Pattern Recognition: Old and New Connections*, I.K. Sethi and A.K. Jain (eds.), Elsevier Science Publishers, pp. 175–194.

Bouchon-Meunier, B. (1991) "Inferences with inaccuracy and uncertainty in expert systems", *Fuzzy Expert System*, A. Kandel (ed.), CRC Press, pp. 43–54.

Bouman, W.J. (1994) *Systeemidentificatie met neural netwerken*, M.Sc. Thesis, Eindhoven University of Technology, The Netherlands.

Chen, S., Cowan, C.F.N., and Grant, P.M. (1991) "Orthogonal Least Squares Learning Algorithm for Radial Basis Function Networks", *IEEE Transactions on Neural Networks*, Vol. 2, No. 2., March, pp. 302–309.

Cohen, P.R. and Feigenbaum, E.A. (1982) *The Handbook of Artificial Intelligence, Volume III*, Addison-Wesley Publishing Company, Inc.

Cybenko, G. (1989) "Approximation by superpositions of a sigmoidal function", *Mathematics of Control, Signals, and Systems*, Vol. 2, pp. 303–314.

Davis, R. (1998) "What Are Intelligence? And Why?", *AI Magazine*, Vol. 19, No. 1, Spring, pp. 91–110.

Giarratano, J. and Riley, G. (1989) *Expert Systems: Principles and Programming*, PWS-KENT Publishing Company, Boston.

Grimaldi, V. (1995) *Perspective d'utilisation des reseaux de neurones non super vises en discrimination des signatures courants de foucault (Using unsupervised neural networks for eddy currents signature discrimination: A prospective study)*, EDF report 95NB00029.

Hameroff, S.R. and Penrose, R. (1996) "Orchestrated Objective Reduction of Quantum Coherence in Brain Microtubules: The 'Orch OR' Model for Consciousness", *Toward a Science of Consciousness: The First Tucson Discussions and Debates*, S.R. Hameroff, A.W. Kaszniak, and A.C. Scott (eds.), MIT Press, Cambridge, pp. 507–540, (http://www.u.arizona.edu/~hameroof/or.html).

Helfman, R., Baur, E., Dumer, J., Hanratty, T., and Ingham, H. (1998) "Turbine Engine Diagnostics (TED): An Expert Diagnostic System for the M1 Abrams Turbine Engine", *AAAI-98/IAAI-98 Proceedings*, AAAI Press, pp. 1032-1038

Jacobs, R.A. (1988) "Increased rates of convergence through learning rate adaptation", *Neural Networks*, Vol. 1, pp. 295–307.

Kamgar-Parsi, B. and Kamgar-Parsi, B. (1990) "Simultaneous fitting of several planes to point sets using neural networks", *Computer Vision, Graphics, and Image Processing*, Vol. 52, pp. 341–359.

Koller, D. (1998) "Structured Probabilistic Models: Bayesian Networks and Beyond", *AAAI-98/IAAI-98 Proceedings*, AAAI Press, pp.1210-1211.

Kohonen, T. (1989) *Self-Organization and Associative Memory*, 3rd ed., Springer-Verlag, Berlin.

van de Kraats, E.J. (1989) "Knowledge-Based Systems: Fit for Purpose", *Non-Destructive Testing: Proceedings of the 12th World Conference on Non-Destructive Testing*, Amsterdam, April 23-28, J. Boogaard and G.M. van Dijk (eds.), Vol. 1, pp. 19–25.

Maes, P. (1994) "Agents that Reduce Work and Information Overload", *Communications of the ACM*, July, Vol. 37, No. 7, pp. 31–40.

Mitchell, T.M. (1997) *Machine Learning*, WCB/McGraw-Hill.

Montague, G.A., Morris, A.J., and Willis, M.J. (1992) "Neural networks (methodologies for process modelling and control)", *Proceedings of the 1992 Symposium on Artificial Intelligence in Real-Time Control*, Delft, preprint, pp. 203–208.

Moulton, M. (1998) "Success Comes From Experiencing Failure...", *Applications and Innovations in Expert systems VI: Proceedings of ES98, the Eighteen Annual International Conference of the British Computer Society Specialist Group on Expert Systems*, R. Milne, A. Macintosh, and M. Bramer (eds.), Springer-Verlag, pp. 263–274.

Nair, S., Udpa, S., and Udpa, L. (1993) "Radial basis functions network for defect sizing", *Review of Progress in Quantitative Nondestructive Evaluation*, Vol. 12A, D.O. Thompson and D.E. Chimenti (eds.), Plenum Publishing, New York, pp. 819–825.

Penrose, R. (1994) *Shadows of the Mind*, Oxford University Press, New York.

Quinlan, J.R. (1993) *C4.5: Programs for Machine Learning*, Morgan Kaufmann Publishers, Inc., San Mateo, CA.

Riesbeck, C.K. and Schank, R.C. (1989) *Inside Case-Based Reasoning*, Hillsdale, Erlbaum.

Rumelhart, D.E., Hinton, G.E., and Williams, R.J. (1986) "Learning internal representations by error backpropagation", *Parallel distributed Processing: Explorations in the Microstructures of Cognition*, Vol. 1, D.E. Rumelhart and J.L. McClelland (eds.), MIT Press, pp. 318–362.

Sarle, W.S. (1994) "Neural Networks and Statistical Models", *Proceedings of the Nineteenth Annual SAS Users Group International Conference*, Carry, NC, SAS Institute, April, pp. 1538–1550.

Searle, J. R. (1980) "Minds, brains and programs", *Behavioral and Brain Sciences*, Vol. 3, pp. 417–457.

Sloman, A. (1994) "Explorations in Design Space", *Proceedings ECAI94, 11th European Conference on Artificial Intelligence*, A.G.Cohn (ed.), John Wiley & Sons, pp. 578–582.

Turing, A.M. (1950) "Computing Machinery and Intelligence", *Mind*, Vol. 59, pp. 433–460

Wang, DeLiang (1993) "Pattern recognition: Neural networks in perspective", *IEEE Expert*, August, pp. 52–60.

Watrous, R.L. (1987) "Learning algorithms for connectionist networks: Applied gradient methods of nonlinear optimization", *IEEE 1st Int. Conf. on Neural Networks, San Diego, 1987, II*, pp. 619–627.

Watson, I. and Marir, F. (1994), "Case-based reasoning: A review", *The Knowledge Engineering Review*, Vol. 9, No. 4, pp. 327–354.

Watson, I. (1998) "Case-Based Reasoning is a Methodology not a Technology", *Research and Development in Expert Systems XV: Proceedings of ES98, the Eighteen Annual International Conference of the British Computer Society Specialist Group on Expert Systems*, R. Milne, A. Macintosh, and M. Bramer (eds.), Springer-Verlag, pp. 213–223.

Zadeh, L.A. (1965) "Fuzzy Sets", *Fuzzy Models For Pattern Recognition: Methods that Search for Structures in Data*, J.C. Bezdek and S.K. Pal (eds.), IEEE Press, pp.35–45. (reprinted from *Inform. Control*, Vol. 8, 1965, pp. 338–353).

Zurada, J.M. (1992) *Introduction to Artificial Neural Systems*, West Publishing Company.

Chapter 4

Case-Based Reasoning

4.1 Introducing CBR

4.1.1 The origins

For years symbolic-AI research has tried to explain the human way of reasoning as chaining together of rules. It was supposed that all the knowledge could be expressed in form of rules, for example, of the *if...then...* kind, and all problems could be solved by applying these rules to a given problem description. This way of explaining reasoning has led to useful results, for instance, in form of expert systems but is now perceived as an incomplete way of describing the human way of reasoning.

End of 1970s, early 1980s, Roger C. Schank has published several articles [Schank & Abelson, 1977; Schank, 1982] proposing a different view on human reasoning. Two important concepts in his view are memory and explanation. Schank suggests that our knowledge about the world is mainly organised as memory packets holding together particular episodes from our lives that were significant enough to be remembered, and if we remembered them it also means that we learned something about them. Schank calls them MOPs (memory organisation packets). The MOPs themselves and their elements are not isolated but are interconnected by explanations of the situations and by our expectations as to the normal progress of events (called scripts by Schank). There is also a hierarchy of MOPs with many bigger MOPs sharing smaller MOPs and with quite complex interconnections.

If a MOP contains a situation where some problem was successfully solved, and a person finds himself in a similar situation that reminds him of the previous experience, he can try to follow the same steps in order to also reach a solution. Thus, problems are solved not by following a general set of rules, but by reapplying previously successful solution schemes in a new but similar context. The important aspect in the memory-based model of reasoning is how the memory packets are retrieved. Generally, we are reminded about something by the similarity, but the retrieval can be also based on differences, as is shown in the MOP example described in the inset. The retrieval is almost never full and is very context dependent. It seems that once we are focused on some MOP it is very

MOP example

Suppose we are driving a car in San Diego, CA, and the fuel is running out, so we do the thing we *always* do *in such situations*, we drive to a filling station. From our *previous* visit to the USA we know that in order to start a pump one has to lift a lever or press a button. We *remember* it because in Europe the pumps are activated automatically and we had to *learn* that it is *different* during our previous visit to the East Coast. However, *this time* the pump does not want to work in spite the fact that we lift a lever, we look then for a button but there is none. From many other situations we know that it does help to ask other people. We are told that one has to pay before one may tank the gas. Immediately, we try to form explanation for this situation recalling one or more of the crime movies we have seen — obviously one has to pay beforehand because they are afraid one will drive away without paying. Now our "US filling station" MOP has been extended with one extra possibility of having to pay before we tank. Because the next several times we have to tank in California we also have to pay before we tank we form a generalisation. Thus there are two scripts for filling up the car in US: one is tank then pay, and another pay then tank. The distinguishing factor, as far as we can tell, is the fact whether we are on the East or West Coast.

easy to recall other MOPS related to it by some features. On the other hand, it may be difficult to just recall all relevant MOPs related to some (especially abstract) term — exhaustive recall is difficult.

The MOP-based model of reasoning may explain some of the difficulties in building rule-based reasoning systems, namely the knowledge elicitation. The experts may forget the specific rules but they do remember the *whole situations* where the rules were applied and were either successful or not [Riesbeck & Schank, 1989]. The skills and knowledge acquired through experience are precisely what makes them experts (if problems could be solved by simply following a set of rules then anybody could solve them). Knowledge elicitation is difficult because of the problems in recalling *all* the relevant rules and/or situations. Generally, people are bad at this type of tasks. If asked, for example, "what can be the reasons for this type of failure", usually, an incomplete list of reasons is given. However, when confronted with the same failure in a real situation, an expert will usually be able to recall all possible causes relevant to the given problem situation. The recall is then done using the whole situation as a starting point, not just an abstract term like "a reason for failure."

4.1.2 Main assumptions and ideas

Case-based reasoning puts the above-mentioned considerations into practice.[1] CBR is in a nutshell reasoning by remembering. In CBR, previously solved problems (called cases) are used to suggest solutions for new problems. This works because, generally, similar problems have similar solutions. Kolodner [1996, p. 359] lists four assumptions about the world that are the basis of the CBR approach[2]:

- *regularity* of the world — the same actions executed under the same conditions will tend to have the same or similar outcomes,
- *typicality* — things and experiences tend to repeat,
- *consistency* — small changes in the situations require only small changes in the interpretation and only small changes in the solution,
- *ease of adaptation* — when things repeat, the differences tend to be small, and the small differences are easy to compensate for.

Figure 4.1 illustrates how the regularity of the world is used to solve problems in CBR. Once a given problem is described in terms used to describe previously solved problems, the most similar, already solved, problem can be found. This gives us an immediate access to the solution of that problem. This solution might be directly applicable to the current problem; however, usually some adaptation will be needed. The adaptation will be based on the differences between the current problem and the problem which served to retrieve the solution.

Figure 4.1 illustrates also the way of solution finding using rule-based reasoning – here, a chain of rules leads directly from the problem to the solution. This might seem simpler than case-based reasoning, but the rules, first of all, have to be known, which can be problematic. Also, the reasoning chain can be very complex and computation intensive (due to backtracking), for example, for planning problems, leading to long processing times. In general, adaptation is simpler than generation of a solution from scratch. Abstracting from the effort needed to construct a system, CBR is preferable to rule-based systems if the sum of effort for case retrieval and adaptation is smaller than the effort for generating a solution from the first principles.

1. In fact, the first CBR system – CYRUS – was developed in 1983 by Janet Kolodner from the Roger Schank's group at Yale University.
2. Statistical classifiers also make use of the assumption of the regularity of the world in the form of assumed distributions from which the observed data originates.

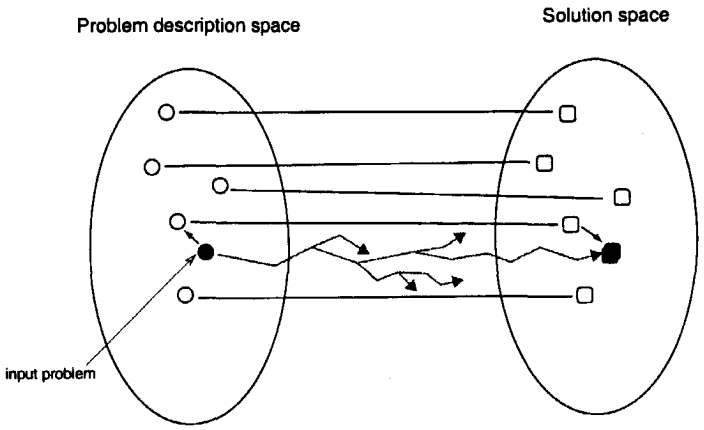


Figure 4.1: Basic idea of problem solving using case-based reasoning. The input problem has to be mapped to the problem description space, a similar solved problem has to be found, and then its solution adapted to solve the current problem. The zigzag arrow illustrates rule-based way of finding a solution. It also shows the abandoned search paths. Solutions obtained using rule-based and case-based reasoning may be not exactly the same. (Figure adapted from [Leake, 1996].)

4.2 CBR working cycles

CBR systems are capable of learning by storing (and subsequent reusing) solutions to previously solved problems. For example, in Figure 4.1, once the solution to the new problem has been verified as correct, a link between it and the problem description will be created and both will be used to solve new problems. Adding new problem-solution pairs (cases) will improve results of the system by filling the problem space more densely.

Solving the problem in Figure 4.1 the following steps were made: description of the problem, search for a similar previously-solved problem, retrieval of the solution to it, adaptation of the solution, verification of the solution, storing of the solved problem. This gives us a working cycle because the newly found solution may be used to solve future problems.

CBR for problem solving. Kolodner [1993] describes the following four stages in a cycle applicable to problem-solving CBR (the cycle is schematically presented in Figure 4.2):

- *Case retrieval* — After the problem situation has been assessed, the best-matching case is searched for in the case-base. This gives us an approximate solution.
- *Case adaptation* — The solution found is adapted to fit the new problem better.
- *Solution evaluation* — The (adapted) solution has to be evaluated. This can be done *before* applying it to solve the problem, and *after* the solution has been applied. If the result was not satisfactory it may be necessary to repeat adaptation or retrieve more cases. Evaluation of the result may lead to learning new knowledge — which can be manifested in changes to indexing, in new adaptation rules, etc.
- *Case-base updating* — If the solution was successful the new case may be added to the case-base.

Processing in real systems may be both simpler, for instance, the adaptation stage can be missing; and more complex, for instance, involving iterative improvement of the solution in either local or global iterations within or between the stages.

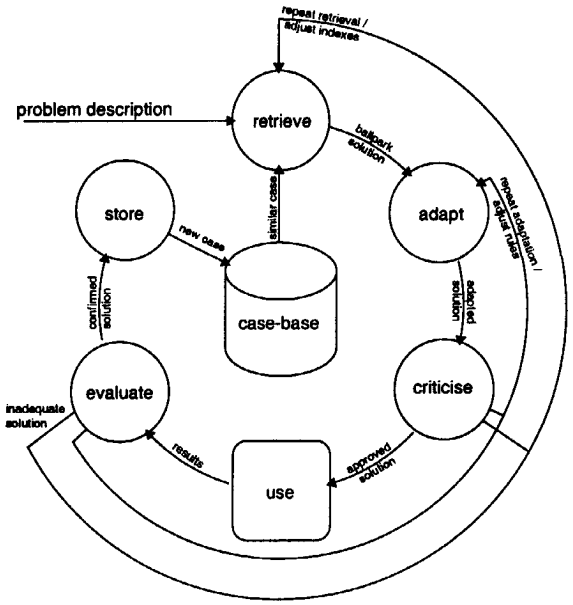


Figure 4.2: CBR working cycle (based on [Kolodner, 1993]).

Aamodt and Plaza [1994] give a slightly different schema of the CBR cycle known as "the four REs," see Figure 4.3. The four REs correspond to the following activities:

- RETRIEVE previously solved case(s) most similar to the current problem,
- REUSE the retrieved case(s) to propose a solution to the problem,
- REVISE the proposed solution if necessary; this can be done by the user of the system or by a set of adaptation rules,
- RETAIN the current case in the case-base if deemed useful.

Though using different terminology, this cycle is similar to that described by Kolodner: the adapt stage is essentially the same as the REUSE stage and the criticise and evaluate stages are combined into a single REVISE step. In Figure 4.3 one can see that the solution can be *applied* to (actually) solve the problem at various stages of the cycle. Where it is done depends much on the characteristics of the problem, mainly on the cost of the solution failing. For example, in an interactive design application it is common to apply the solution and then evaluate the results and revise it. In an application where a failure is expensive (like air traffic control [Bonzano, 1998]) the evaluation and the revision of the proposed solution are done by, for example, an operator before the solution is applied. Usually, the results will be evaluated before the case is stored into the case base. This is made explicit by Aha [1997] by using REVIEW stage after REVISE (a "five REs" cycle) and by Thompson [1997] by using two REVISE stages in the CBR cycle, one before and one after the application of solution.

CBR for interpretation. The above presented CBR cycles are generally applicable to problem-solving CBR. The situation is a bit different when CBR is used for interpretation. First, there is usually less adaptation done, if adaptation is done at all. While, generally, a solution to a problem may take various forms and thus allows various adaptation possibilities (e.g., in the meal planner JULIA described in [Kolodner, 1993]), the interpretation (in NDT usually consisting of classification and

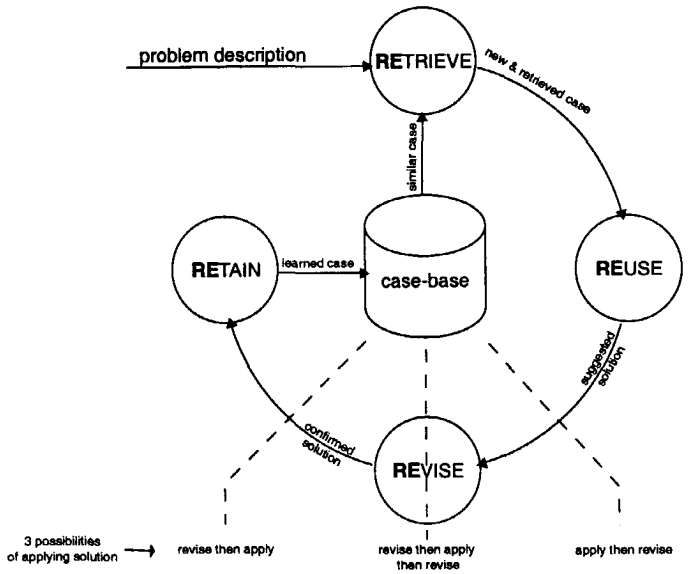


Figure 4.3: The 4 REs CBR cycle (adapted from [Aamodt & Plaza, 1994]).

characterisation) leaves much less room for adaptation (e.g., a medical diagnosis/classification system PROTOS [Porter et al., 1990] does no adaptation). What becomes important is comparison between the current situation (or data) and the retrieved one(s). Figure 4.4 shows a schema of the working cycle for case-based interpretation:

- first, situation assessment is done, usually, this entails extraction of relevant features describing the problem (see section 4.4.),
- next the most similar case or cases is/are retrieved from the case-base,
- after analysing the match between the current and the retrieved case(s), an interpretation is made, either automatically or with a help of the user of the system,
- if the new problem is significantly different it is stored as a new case in the case-base.

What can be directly noticed is that the adaptation stage is not explicitly mentioned in the cycle. Some necessary adaptation may be present but usually the interpretation is the same as for one of the retrieved cases. What takes a prominent role is situation assessment (which is generally more complex than in the case of problem solving) and the comparison and match evaluation. It is necessary to determine whether the same interpretation can be applied to the current case as it was done for the retrieved case(s). The prototype systems described in Chapters 6 and 7 use this type or reasoning cycle for NDT data interpretation.

CBR cycles are rarely done completely without human intervention. Often adaptation, revision, and/or evaluation are done by a human, usually, by the direct user of the system. If the user is not proficient enough in the problem domain, some more difficult cases can be handled by a domain expert, for example, case storage can be postponed until the new cases have been analysed by the expert. CBR systems are usually designed not to replace humans in problem solving but to assist them.

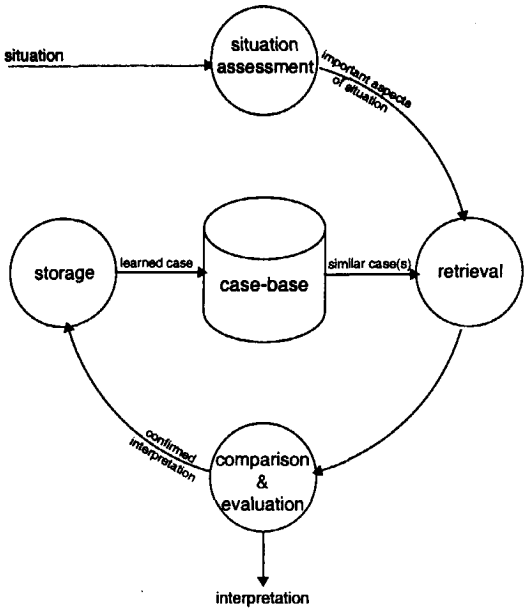


Figure 4.4: Working cycle for case-based interpretation (based on [Leake, 1996, p. 7]).

4.3 Use of multiple types of knowledge in CBR

CBR systems make use of many types of knowledge about the problem domain they are made for. Some of this knowledge can be explicit as, for instance, in adaptation rules, and some less apparent. The power of CBR systems lies in their ability to increase their knowledge by collecting solved cases, however, simply collecting more data does not necessary make the system more intelligent. The data has to be properly interpreted by the system.

Richter [1995] identifies four knowledge containers present in CBR: the vocabulary (case description), similarity measures, adaptation knowledge, and, of course, the cases themselves. The first three containers contain compiled knowledge, and the cases contain knowledge interpreted at the run time [Wilke et al., 1997]. The compiled knowledge usually represents the general knowledge about the problem domain. If there are any exceptions from this knowledge they are typically handled by appropriate cases. The types of knowledge that can be found in the four containers are presented in the following:

- *Vocabulary* — This container includes the knowledge needed to choose the features used to describe the cases, but also knowledge needed in the design of the situation assessment stage:
  - *Case features* — have to be chosen so that they can be helpful in finding other cases which contain useful solutions to similar problems and, at the same time, they have to be discriminative enough to prevent selection of too different cases, which could lead to false solutions or reduce performance. Good knowledge of the problem domain is necessary to choose from all the problem parameters these which are best as case features. For the sake of efficiency it is good to limit the number of features used. For further discussion of case features see Section 4.7.1.

  - *Situation assessment (elaboration)* — is used to extract the case features from the raw problem description. In some situations it requires preprocessing of the data describing the problem. This stage can be used to remove or compensate for these parameters which are irrelevant to the problem to be solved, this can simplify further processing in the system. Of course knowledge is required how to do it and how to do it reliably.
- *Similarity measures* — This container includes the similarity measure itself, but as the main goal of the similarity measure is retrieval, this container can be considered to comprise also the knowledge used to choose the case-base organisation and the method of case retrieval.
  - *Similarity measure* — The similarity measure that can be used in a given system is constrained by the chosen case vocabulary. Still, for any given problem there are many possible similarity measures that can be used. Even for a simple distance between numerical feature vectors there are many possibilities. The situation becomes more complex if feature weights are used or cases have complex structure. Especially, for complex structured cases and for interpretation/classification problems (where the value of the similarity can be used as basis of automatic classification) implementing the similarity measure requires good knowledge of the problem domain.
  - *Case-base organisation* — Knowledge about cases can be used to organise the case-base structure so that the cases can be accurately and efficiently retrieved. Often automatic algorithms can be used to build the index structure (e.g., a decision tree), but sometimes a "hand-crafted" case-base hierarchy based on the knowledge of the problem domain can be more efficient.
  - *Retrieval* — is largely determined by the chosen case-base organisation and similarity measure, but still there are many choices, for example, about the way of backtracking if no good case was initially found, which require domain knowledge.
- *Adaptation knowledge* — This is the most obvious type of knowledge in a CBR system, maybe because it is usually expressed in a form of rules. Case-based interpretation systems usually do not have an adaptation stage, however, their evaluation and/or argumentation stages also use knowledge which fits into this knowledge container.
  - *Adaptation* — Generally, adaptation requires knowledge about how differences in the problem affect the solutions. Often, the knowledge used is a partial domain model. This knowledge will usually be coded in explicit rules. For many problem domains this is the most difficult knowledge to acquire. Therefore, frequently, the adaptation is left to the user of the system.
  - *Evaluation and argumentation* — as done in interpretative CBR have to use knowledge about significance of the differences and similarities between the situations. It can be considered to extend the knowledge contained in the similarity measure; however, because it is applied to only a selection of cases in can be more extensive (time is not so critical).
- *Cases* — typically contain solved problem instances and in many systems represent the knowledge that the system acquires during use. What the cases will contain is mainly determined by the chosen vocabulary. Some CBR systems are delivered with an initial case-base which contains carefully selected cases that provide as even as possible coverage of the problem domain. New cases will usually be added during use and are the main way of acquiring new knowledge. However, often, it is unwise to store all the solved problems as cases. In such situations one has to specify heuristics used to determine the useful cases which will be stored in the case-base.
- *Other knowledge types* — Apart from the four containers also other knowledge types may be incorporated into a CBR system. An example is the proper design of the user interface. Many systems leave adaptation, evaluation, and/or revision to the user. This has to be done as effi-

ciently as possible and requires knowledge about best ways of presenting the cases to the operator — many problem domains have a typical way of problem presentation familiar to the users.

Some of this knowledge can be obtained by analysing the available data (cases) using, for instance, statistical methods or methods from machine learning (Section 3.5.2), like decision trees for case-base organisation (Section 7.5.5) and rule induction for adaptation knowledge [Wilke et al., 1997; Hanney & Keane, 1997]. However, exclusive use of such techniques may result in a system unprepared for handling situations significantly different from the available data. This can be properly handled if also knowledge about the problem domain as, for example, possessed by the expert is used. One can also use automatic tools for the validation and refinement of the knowledge present in a CBR system, as indicated in [Craw, 1998].

The fact that there are so many knowledge types in a CBR system means that during the design phase decisions have to be made how the knowledge will be distributed among the knowledge containers. One can, for example, decide to put more effort into collecting a large set of cases to keep the necessary adaptation stage simple, or on the contrary, to use a good domain model to build a powerful adaptation stage and to keep the number of cases small. Another choice that can be made is the balance between similarity measure and the subsequent evaluation rules. One can consider the whole problem of knowledge distribution as an optimisation problem given the intended application domain and application circumstances of the CBR system.

CBR systems are capable of learning, therefore, apart from the knowledge built into the system during the construction, they will acquire new knowledge during use. They can learn both from the successes (usually just by storing the cases) and from the failures (e.g., by additionally adapting the similarity measures). Usually, the failures have to be explained to the system by the users, as this is, for instance, done in PROTOS [Porter et al., 1990].

4.4 Advantages of CBR

CBR has in recent years become widely popular, this is because it has many advantages. Aamodt [1993] even suggests CBR as an answer to many problems of knowledge-based systems — case-based methodology combined with general models can lead to more robust, adaptive systems having high degree of interactivity with the user and the environment. Some of the more specific advantages of CBR are listed below (based on [Mark et al., 1996; Kolodner, 1993]).

Efficiency in solving problems. Rule-based systems solve problems by starting from a problem description and following a chain of rules. These derivations can be time consuming and sometimes not thoroughly understood. A CBR system begins its reasoning with an already solved similar problem, therefore, it can be expected to be more efficient.

Ability to learn. Probably the most attractive facet of CBR systems is their ability to learn. This means that a CBR system can be put into operation with a minimal set of solved cases in the case-base, and fill it with new cases as it is used, this way increasing its problem solving ability. Apart from simply adding new cases, new indexes in the case-base can be created or existing indexes can be changed.

Similar to other instance-based, lazy problem-solving methods CBR systems are capable of online (incremental) learning [Aha et al., 1991]. New cases can be simply added to the case-base, and the actual use of the information they contain is deferred until new problems have to be solved. This is in contrast to many Machine Learning methods (see Section 3.5.2) which require a special period of training when information extraction (knowledge generalisation) is performed. This and other characteristics of CBR as a lazy problem-solving method are further discussed in Section 4.6.

Use in not-fully-formalised (weak-theory) domains. Not fully formalised domains are characterised by incomplete understanding of the domain, by problems with representing the domain using formal languages (e.g., rules), or by the sheer volume of the knowledge in the domain. CBR does not depend on fully modelling this knowledge, instead, it needs knowledge of how to recall and use previous cases. Still, the results can be reliable, because CBR can rely on what has worked in the past. CBR systems can make use of the "experience" part of the expertise, while the expert systems concentrate mainly on the "knowledge" part of the expertise, which in many domains may be very small. Thus, CBR systems can be used to solve problems in domains for which no models exist (e.g., CLAVIER – Section 4.12.2).

Reduced construction costs, shorter building times. This applies mainly to situations where the available knowledge is "naturally" structured as cases, which simplifies the problem of knowledge acquisition. Compared to statistical classifiers the initial construction costs are higher but later use can be less expensive because CBR systems can adapt to changing conditions (also on-line, during use), while statistical classifiers may have to be retrained. A CBR system will generally be more flexible (adaptable) and thus suitable for a larger domain of application. Moreover, CBR systems can often be developed incrementally, because already simple retrieval facilities may turn out to be useful for the users.

Simplified knowledge acquisition. Traditionally, an expert system is built on the basis of an expertise of a person who is an expert in solving some problem. The formalisation of the knowledge of an expert may be difficult, especially when the problem domain itself is complex but also when knowledge is characterised by many exceptions from general rules. Both situations can be usually better handled by CBR systems. In fact, expertise may be more useful for building a CBR system than for an expert system. Generally, the knowledge required for CBR will not be that deep, thus easier to formalise. Moreover, knowledge about the adaptation of solution is usually much easier to acquire than knowledge about the derivation of solution. Also, knowledge about adaptation can be better structured based on the types of problems (cases) it concerns. As far as exceptions are concerned, they can be handled by appropriate cases. In an ideal situation, elicitation becomes largely a task of gathering cases and identifying significant features that describe them.

Increased reliability. Because CBR systems are designed to recognise when new problems are encountered they should be more reliable than, for example, simple statistical classifiers. CBR systems are built not only based on the data, as the majority of other classifiers are, but can also make use of available a-priori knowledge about the problem domain.

CBR is good at handling exceptions. It does not make the closed-world assumption of many rule-based (model-based) systems or make wide generalisations as some classifiers (e.g., ANNs) do. By using cases, the knowledge is contextualised which results in robust systems [Aamodt, 1993] — if generalisations are made, they are made within a specific context. Being an instance-based technique and storing the cases, a CBR system is able to determine the confidence in the solution based on the distance to the neighbouring cases in the case-base (cases define areas of applicability of knowledge). CBR is explicit about not exploiting the whole problem space, while, for example, expert systems may hide their inability to solve all the problems. Moreover, CBR can recall previous problem areas and avoid repeating mistakes. Of course, all of the above holds only if the assumptions listed in Section 4.1.2 are true.

Ease of maintenance compared to rule-based systems. This is partly due to the fact that CBR systems can adapt to many changes in the problem domain just by acquiring new cases. This eliminates some need for maintenance. Also, because generally the rules present in a CBR system, for instance, for adaptation or matching, are simpler than in a fully rule-based system, they are easier to maintain. Usually, however, there is little need to adjust the rules and the need for maintenance is confined to maintaining the case-base(s).

Easier to explain and justify the results. The results of a CBR system can be justified based on the similarity of the current problem to the retrieved case(s). Because solutions generated by CBR are easily traceable to precedent cases, it is easier to analyse failures of the system. This is especially important if a system failure may result in legal consequences. In a rule-base system the whole reasoning chain has to be reconstructed and analysed which may be difficult.

Better acceptance by the users. The reasons for this are twofold. First, the users have usually a better understanding of the working of a CBR system than would be the case with a rule-based system (this is related to the previous point). Second, CBR systems are usually constructed in a way that includes the user in the decision-making process. Most of the CBR systems operate in a close cooperation with a human operator, in fact, this is almost a necessity unless the rules used in the adaptation phase are powerful, or the problem domain is simple.

Corporate knowledge accumulation. The acquired cases may be used also for purposes other than problem solving only. They reflect the organisation's experience and can be used, for instance, for personnel training (computer assisted learning), optimising production procedures, etc.

Scales better up than neural networks or expert systems. ANN may have difficulties in learning complex problems (e.g., going from a 3- to 6-degree-of-freedom robot), and complex rule-bases in expert systems are difficult to manage. If a more complex problem can be solved by using more cases then this may only pose hardware problems (hardware too slow, large memory requirements) but not a conceptual problem. (However, the number of needed cases may increase stronger than linearly with increasing problem complexity.)

Handling of uncertainty. CBR systems can in principle handle uncertainty well. For example, the measure of similarity of the retrieved case and the amount of adaptation done may provide information about the certainty of solution.

4.5 Disadvantages of CBR

Like any technique CBR also has its disadvantages. The most important are listed below.

Handling large case-bases. CBR systems using large case-bases may have high memory/storage requirements and may suffer from long retrieval times. However, the order of both is at most linear with the number of cases, and the problem becomes less and less significant as the computer hardware gets faster and memory prices fall down (see Appendix A).

Dynamic domains. CBR systems may have problems in dynamic domains where they may be unable to follow a shift in the way problems are solved (not enough context information, reuse of outdated cases). An example of such a domain is optimal PC configuration — what was good half a year ago may already be outdated.

Bias towards the past. Solutions provided by CBR systems may be too strongly biased towards what already has worked. This can be a disadvantage, for example in design. CBR systems are not creative and they can also bias a human user to too much reuse of old solutions.[1]

Difficulties in handling noise. Parts of the problem situation may be irrelevant to the problem itself. Sometimes these elements take a form of noise. If the noise is not recognised in the situation assessment stage and becomes part of the problem description then this may result in the same problem unnecessarily being stored numerous times in the case-base because of the differences due

1. Lack of creativity is common to computer systems. Already in 1842 Lady Lovelace wrote: "The [Babbage's] Analytical Engine has no pretensions to *originate* anything. It can do *whatever we know how to order it* to perform" (quoted after Turing [1950]).

to noise. This reduces the performance of the systems. The problem will be more pronounced in CBR systems that work with multimedia data and/or acquire data directly from the environment.

Fully automatic operation. CBR systems are learning systems; this means that some problem situations can occur for which the system has no (good) solution. A CBR system then usually expects input from the user, unless it can fall back on a large knowledge-based or algorithmic system that can solve such problems (in such situations CBR would be used mainly because of faster speed).

4.6 CBR as a lazy problem-solving method

CBR is a lazy problem-solving method, therefore, it shares many characteristics, including advantages and disadvantages, with other lazy problem-solving methods, which are common in pattern recognition and Machine Learning. Aha [1997] defines lazy problem-solving methods as these which display the following behaviour characteristics (three Ds):

- *Defer* — They do not process the data they contain until given an information request.
- *Data-driven* — They respond to request by combining information from the stored data.
- *Discard* — They dismiss any temporary intermediate results obtained during problem solving.

The opposite of lazy problem solvers are eager algorithms which try to extract as much information from the data as possible prior to the actual problem solving and to discard the data. An example of a lazy problem solver is a k-nearest-neighbour classifier, while a decision-tree classifier is an example of an eager problem solver. One often describes eager algorithms as performing knowledge compilation, while the lazy algorithms do run-time knowledge interpretation. The knowledge compilation has usually the form of generalisation. A major disadvantage of generalisation from the data is that it usually means loss of information.

Aha [1997] lists the following benefits of lazy problem-solving methods:

- *Ease of knowledge elicitation* — Lazy methods can make use of more easily available cases, examples, or problem instances instead of difficult to extract rules. This means that knowledge acquisition can be refocused on case acquisition and structuring.
- *Absence of problem solving bias* — Because cases are in raw form they can be used for several different problem-solving purposes. In contrast eager methods can be used only for the purpose with which the knowledge compilation was done.
- *Incremental learning* — The training costs are typically low, as they can often be confined to storage of new cases. Dynamic (on-line) adaptation to changing environment is possible [Fisher & Schlimmer, 1988].
- *Suitability for disjunctive solution spaces* — Lazy approaches are often more appropriate for complex solution spaces, which could be difficult to handle for eager approaches which replace data with abstractions obtained by generalisation.
- *Ease of explanation* — Because original case data is stored, lazy approaches can easily provide precedent explanations. More abstract explanation can be derived by analysing the stored.
- *Suitability for sequential problem solving* — Sequential tasks, like these encountered in reinforcement learning problems, often benefit from the storage of a history in the form of a sequence of states. Lazy approaches can easily do it.
- *Highly intuitive* — Experts often describe their problem solving behaviour in a way that suggests a form of case-based reasoning.

Major disadvantages of lazy problem solvers are their memory requirements and slower execution speed because of more work necessary to answer the queries. The lazy and the eager techniques can be considered as the opposite edges of a whole spectrum of techniques[1], with many intermediate

1. One might argue that pure lazy methods do not exist, as any acquired data is already a knowledge compilation.

problems solver types [Aha, 1998]. There may be eager variations of lazy methods, which try to circumvent some memory and speed problems, or lazy variations of eager problems, for example, decision-tree induction in response to a query. One can also integrate lazy and eager approaches in one system like has been done in the INRECA (Induction and Reasoning from Cases) system [Althoff et al., 1996].

4.7 Cases

Cases are the basis of any CBR system. A system without cases would not be a case-based system. On the other hand, a system using only cases (especially simple ones) and no other explicit knowledge (not even in the similarity measure) is in principle difficult to distinguish from a nearest-neighbour classifier or a database retrieval system.

Every case has a certain problem coverage, that is a range of problems that can be solved based on a given case. The coverage depends on the problem type, on the particular case (general vs. exceptional cases), and, to a large extent, on the adaptation facilities (system with no adaptation will have a small case coverage, while a system with very extensive adaptation needs few cases to cover large problem space).

Cases are usually concrete (recording parameters of actually solved problems), but abstract cases can also be used if they can easily be made useful for a given situation (for example just by reinstantiation). Very abstract, all encompassing, cases are no cases at all, as cases are characterised by their "chunkiness."

4.7.1 Case representation

A case consists at least of the problem description and of the solution to that problem. Kolodner [1993] names these two components the *context* and the *lesson*. This is a more general description, because, for example, a lesson may also include information that in a given situation (context) no solution could be found. Apart from these two components, more complex CBR systems may also store additional information with a case, like: the outcome of applying the solution, evaluation of the solution (essential to learning), reasons for a failure (if such occurred), the derivation of the solution, pointers to similar cases, etc. The more information is stored, the more useful the case can be. On the other hand, entering all the information may make the system difficult to use, and actually using this information makes the system more complex. Probably because of these reasons most of the CBR systems are limited to storing only problem descriptions and solutions.

It is useful to store the retrieval statistics and other accounting information for each case. They may include: the number of times the case was retrieved, the average match value, the count of matches above some threshold, etc. These statistics may be used to guide the retrieval, for instance, by prioritising cases; to prune the case-base, for instance, by removing seldom used cases; and, generally, in the case-base maintenance.

Problem description. Problem description essentially contains three components: the situation description, the goal (what is the desired outcome of the solution), and the constraints on these goals. In many systems the constraints and the goals are system-wide (the same for all the cases) that is why only the situation description is actually stored.

Two types of problem description can be distinguished — raw and parametric:

- *Raw problem description* — contains as much original data about the problem and its context as possible. It is mainly used to reconstruct the original situation, for example, when one wants to verify the correctness of the system as a whole or to justify a particular solution. The raw problem description may contain a lot of data and it may be efficient to store it in a separate database.

- *Parametric problem description* — used by the CBR system to solve the problem (important for retrieval and adaptation). In principle it can be derived from the raw description, but derivation can often be time consuming or requires human assistance to interpret the data; therefore, the parametric problem description is often entered by the users when defining a new case. In some situations the parametric problem description is sufficient to fully describe the problem, so there is no need to store the raw problem description.

Deriving a problem description is called elaboration or situation assessment. Sometimes, before the transition from the raw to the parametrised description can be made, the extent of the raw description has to be delimited, for instance, in the LISSA system, described in Chapter 6, this would correspond to the detection phase.

Problem parametrisation can be easy when, for instance, the problem already is in an attribute-value form like in the case of database records; but it can also be complex, for example, it may include complicated algorithmic processing or require a separate knowledge-based system. For example, the CBR system described in Chapter 7 uses a rule-based system to recognise bolt-holes in sections of the images to be classified. Typical examples of some difficult parametrisation problems are:

- *Design* — The difficulty is in the specification of all significant factors, some of them may become obvious only when the solution has been critically evaluated.
- *Text documents* — CBR systems often work with text data, for example many help-desk systems. Text is usually compared based on the word statistics but ideally one would want to extract the meaning from the text (the NL problem).
- *Images* — Here the situation is similar as with text, the retrieval can be done without [Coulon, 1993] or with image understanding. Image understanding usually implies segmenting the image into meaningful elements and describing their relations (see, e.g., [Perner, 1998] and Section 7.4).

An important task of situation assessment is recognition of noise in the problem and its removal. If noise is included as part of the problem description then it will significantly reduce system performance, because by definition noise is not repeatable and noisy features will rarely match with other cases, reducing probability of retrieval of a good case.

The parameters (vocabulary) used to describe cases should be sufficient to describe all the already known cases. However, because CBR systems learn, the vocabulary should be chosen anticipating future expansion or the system should make the expansion of the vocabulary (creating new indexes) easy. Otherwise, it may be impossible to represent new problem features. These new features may get mapped to the available descriptors or ignored, in both cases possibly leading to finding wrong solutions.

Problem solution. The problem solution can be either atomic or compound. An atomic solution is typical for CBR systems used for classification and diagnosis. Some atomic solutions may have fixed action sequences (procedures) assigned to them that can provide finer parameters of the solution (e.g., in LISSA described in Chapter 6 a solution consists of a classification and of a characterisation determined using a procedure attached to the case). Compound solutions can be found, for example, in CBR systems used for planning or design. A compound solution may be composed of a sequence of actions, an arrangement of components, etc.

The main use of a solution is to serve as a starting point for the derivation of new solutions. Therefore, the way a solution was derived may be equally important as the solution itself, especially when generative adaptation is used (Section 4.9.1). If this is the case then it is necessary to store the reasoning steps used to solve the problem (and possibly also justifications for these steps). These may include pointers to other solutions (cases) which were considered but not chosen, either being less suitable or not acceptable at all.

Representation formalisms. Cases can be represented using a variety of formalisms beginning from simple feature vectors to complex networked representations, possibly intertwined in a dynamic memory structure (see Section 4.8.1) or even directly using the Schank's (see Section 4.1.1) idea of MOPs and scripts as, for instance, in [Watson & Oliveira, 1998].

Cases can be stored as an unstructured collection of parameters or they can have a structure. Simple use of structure is to organise a complex case like the B-scan image in Section 7.5.3. A more sophisticated use is to let various cases share parts of their structure thus making use of their similarity. Structured cases are often represented in an object oriented way. Use of inheritance simplifies entering new cases. Inheritance can also be utilised in calculating similarities [Bergmann & Stahl, 1998].

Cases can be monolithic or compound. Individual parts of compound cases can be processed or used separately. For example, a problem can be solved by reusing partial solutions from several compound cases, like in planning or design where the solutions are often compound.

Most representation methods for the more complex cases are proprietary; however, research has been dane on standards suitable for portable case representation, for instance, CASUEL [1994]. Portable representation may simplify sharing of cases between applications — after all, cases are knowledge and knowledge is money.

Visualisation. Another issue is case presentation, that is displaying its information in a form most useful for the users. This is, for example, important in applications of CBR in design or training. Symbolic case representations used internally by a system may be difficult to understand by the users of the system. When cases are retrieved and presented using these symbolic representations, the user may have difficulties in estimating their usefulness for solving a given problem. Often, a visual presentation is more suitable than the symbolic one. This is easy if a case was derived from a visual representation — easy access to the raw case description can be here useful. Still, this will work only for the original cases. If, as a result of adaptation, the case description has changed then the raw case data (e.g., a photograph or a drawing) cannot be used to visualise the new solution. Sometimes, for instance for certain design applications, one can work with schematic drawings where there is a clear mapping from the schematics used to the corresponding symbolic representation and vice-versa. Watson & Oliveira [1998] suggest use of virtual reality environment for the representation and visualisation of cases — they give an example of CBR training application. A VR environment provides visualisation of original and adapted cases "for free."

4.7.2 Indexing

Within the CBR community there has been some confusion as to what indexing actually means. Kolodner [1996] writes that indexing is essentially a historically originated term for what is an *accessibility problem*. Indexing refers to *"the whole set of issues inherent in setting up a memory and its retrieval processes so that the right cases are retrieved at the right times"* (p. 355), thus it defines essentially the scheme for retrieval of cases from the case-base. Proper indexing will guarantee that the case retrieval will (efficiently) return cases most useful for problem solution.

In principle, an index can be just a pointer to a case, but usually indexing is much more than this. Accessibility involves *"a combination of processes and representations like: situation assessment, recognition of the important features of the case, search functions, similarity metrics, vocabulary and matching functions"* (p. 355). Different implementations may make different use of these processes. In a simple implementation, indexing may be synonymous with the parameters describing the problem. In fact, often, by case indexes one understands a selection of case features used for retrieval purposes. Flat (parallel) search scheme for retrieval is primarily dependent on the matching procedures. In more complex systems, indexing may comprise the whole architecture of the case-base.

If we consider the meaning of indexes as these case features (labels) that are used by the retrieval algorithm to retrieve useful cases, there rises a question as to which are the best features to use. The good indexes should be: predictive of the case relevance, recognisable (it should be understandable why they are used), discriminative, but also provide coverage (should not be unique to just one problem situation), etc. These characteristics are the good ones from the conceptual point of view. From the implementation point of view, one would prefer the use of easily available, cheap to compute, surface features as these facilitate fast searching. However, use of only such features without looking at, for example, their predictive capability, may result in a failure to retrieve relevant cases.

Techniques are available (e.g., inductive learning as used in decision trees) which can be used for the automatic choice of the indexes. However, they may choose indexes based more on the training set of cases used for induction than on the real problem (compare Section 5.3.4). Kolodner [1993, p. 281] writes that people tend to be better at choosing the indices than the automatic algorithms. People have a better knowledge of the domain — automatic algorithms may use no domain knowledge at all. Usually, the best indexing vocabulary is drawn from the concepts that naturally come up during carrying out the reasoning tasks.

Indexes do not have to be static, but may change during use. In fact, changing the indexes is one way how a CBR system may learn. Changes may be made if, for instance, a wrong case was retrieved. These changes may involve changing weights (importances) of the features, adding new features to case description, changing or adding pointers to other similar or different cases in the case-base. Some of the changes may be done automatically, so that the failure will be avoided in the future. Changes can also be done based on an explanation of a failure provided by the user (expert) like done in PROTOS [Porter et al., 1990].

4.7.3 Matching — the similarity measure

Case matching is done to determine the similarity between cases. As already mentioned in Section 4.3, the similarity measure contains in some CBR systems a significant part of the knowledge about the problem domain. One reason for this is that one is interested in "useful" similarity, i.e., such that helps to find cases useful for solving the problem. For example, in LISSA (Chapter 6) a transformation-invariant similarity measure is used because the transformation parameters are irrelevant to the class of the signal.

If a problem is described using a number of features, then usually some form of a distance measure between the feature vectors is calculated. Though this might seem simple it still involves many not always obvious decisions like: what distance measure should be used (Euclidean, correlation); if the features are not equally important then what is the importance of the individual features; what should be done with the features missing in the problem description; how to handle combinations of Boolean, discrete, and continuous features; etc. (A discussion of the issues involved in calculating distances between feature vectors can be found in [Wilson & Martinez, 1997].) Often, distinction between local and global similarities is made: the local ones correspond to similarities between individual case features, and the global ones to the way local similarities are combined.

If a problem is described using free-form text then, ideally, matching should be done after the meaning of the text has been determined. However, many successful CBR systems do text matching using statistical techniques, based for example on (co)occurrence of words, pieces of words (e.g., trigrams – see CBR-Express in Section 4.12.3), etc.

For some problem domains, like in design and for images, matching can be very complex. Here, the matching is not only difficult from the implementation point of view, but it is difficult also conceptually. The difficulty becomes obvious when one has to choose the best from a several possibilities, as the notion of similarity is in many situations subjective. Because, often, judging the appropriateness of a solution is easier than comparing the problem descriptions, one can partly

solve this problem in CBR by retrieving several best matching cases, use them all to find possible solutions, and then choose the best solution (and not the best matching problem).

The time necessary to do the matching is sometimes the most important factor determining the retrieval speed. Therefore, reducing the time-complexity of the matching algorithm will usually increase the speed of a CBR system. If calculating a match involves several steps (e.g., matching several subproblems), then the retrieval can be speeded up if the matching can be broken off when it becomes clear that the match will be worse than some best match so far.

4.8 Case-base: organisation & case retrieval

The organisation of a case-base and the retrieval algorithm are related to each other. Case-base should be organised in a way that facilitates accurate and efficient retrieval. Accurate retrieval guarantees that the most appropriate (most similar) case will be retrieved. This might be impossible if, for example, a case-base had a structure making some cases inaccessible to the retrieval algorithm. Efficient retrieval guarantees that cases will be retrieved fast enough to give acceptable system response times. These two factors are related: it is easy to guarantee accurate retrieval at the expense of efficiency (e.g., by matching all the cases) and is easy to have fast retrieval if only a fraction of the case-base is searched, possibly missing some relevant cases. Therefore, a good case-base organisation and a good retrieval algorithm are such which achieve the best compromise between accuracy and efficiency.

4.8.1 Organisation

In this section several possible methods of case-base organisation are given. Also a combination of these methods in one case-base is possible as, for example, described in Section 7.5.5.

Flat organisation. The simplest case-base organisation is a straightforward flat structure. During retrieval, in principle, the whole case-base has to be searched and every case matched. For large case bases with complex case representations this can result in long retrieval times. The retrieval can be speeded up by a use of faster hardware and/or a parallel search. Also, for some problems, a shallow indexing scheme may be used to select candidates for retrieval, for instance, one can select all cases sharing some descriptor.

The advantages of the flat organisation are, for example, its simplicity, ease of case addition and deletion, insensitivity to a use of dynamic similarity functions, which might require restructuring of the more complex case-base configurations.

Clustered organisation. This type of organisation stores cases in clusters (groups) of similar cases. The grouping of the cases may be based on their mutual similarity, like in the case-base described in Section 7.5.5, or based on the similarity to some prototypical cases as described in [Malek, 1995; Malek et al., 1998; Schmidt & Gierl, 1998]. Each cluster is represented by some prototypical case, either within the cluster or outside of it, either a real case or some abstract case. The advantage of this organisation is that it is easy to select the clusters that will be matched based on the values of matches with the prototypical cases. Also, it requires little additional knowledge about the problem domain apart from the similarity measure.

A disadvantage is a larger cost of case addition and deletion. This type of organisation is thus more suitable for static case-bases (it is unsuitable for CBR systems with dynamically changing similarity measures).

Hierarchical organisation. A hierarchical structure provides both a way of finding the right case and a way of speeding the retrieval. A disadvantage is the higher complexity and the cost of adding or removing the cases. Hierarchical networks can be built with preassigned importances attributed

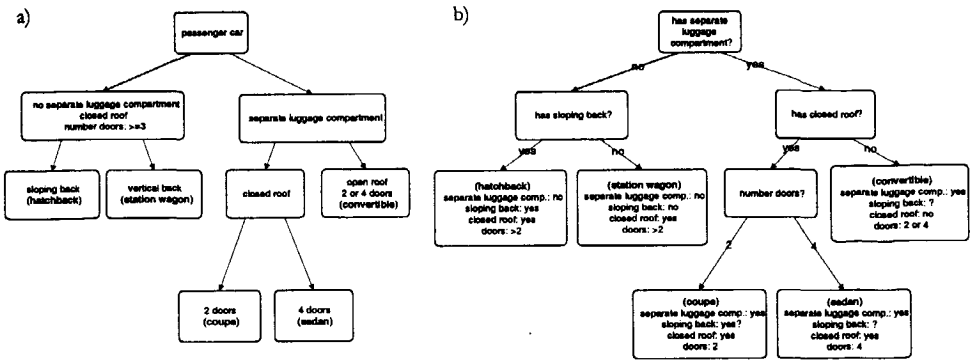


Figure 4.5: Shared feature network (a) and discrimination network (b). These example networks have similar structure but this is not always the case. The shared feature network stores data in the network, while the discrimination network stores the data in the nodes. (In real world the distinction between coupes and sedans, as well as hatchbacks and station wagons can be ambiguous).

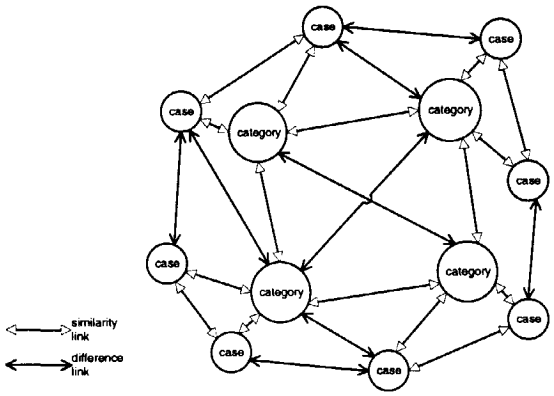


Figure 4.6: Networked case-base organisation — category-exemplar model.

to the particular features. However, this importance assignment is static. If it changes then the hierarchy has to be reorganised. Many possible ways of hierarchical organization exist, for example:

- *Shared feature networks* — In these type of networks the hierarchical organisation is obtained by grouping together cases which share the same features. An example is shown in Figure 4.5a.
- *Discrimination networks* — These are based, for example, on a decision tree (e.g., ID3, C4.5 [Quinlan, 1993]) where a three induction algorithm is used to determine the discriminating features, see Figure 4.5b.

Networked organisation. An even more complex organisation is built as a network of categories, cases (category examples), index pointers and relations, see Figure 4.6. This type of organisation is, for instance, used in PROTOS. The cases can be associated with the correct categories and with other similar cases using similarity links. During retrieval, cases are, first of all, retrieved based on the similarities. If use of such similar cases leads to failed solutions, then difference links are established to the incorrectly matched cases. The difference links are assigned based on the failure explanations provided by the user. This type of case-base organisation is useful not only for the retrieval but also for reasoning about the cases, as it represents various relations between cases and categories.

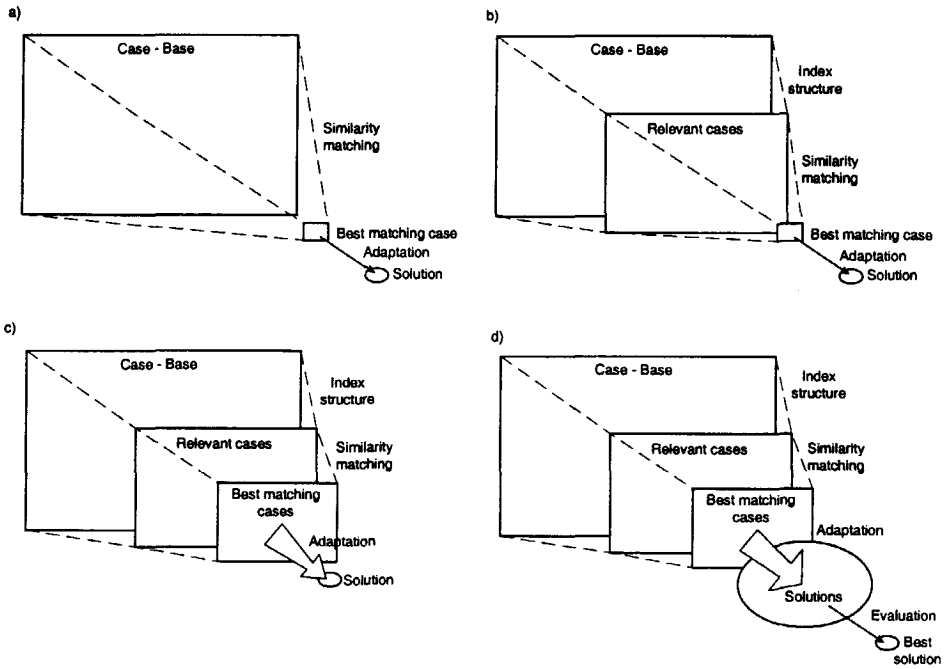


Figure 4.7: Various possibilities of retrieval: a) flat search, b) selection and similarity matching, c) deriving a single solution from a number of cases, d) deriving a number of solutions and choosing the best one.

4.8.2 Retrieval

Retrieval is the primary process in CBR. In fact, many CBR systems are just intelligent information retrieval systems. At first, it might seem that one could use standard database techniques for the retrieval, like query by example; however, CBR needs to be able to retrieve partial (inexact) matches and the standard provisions for partial matching available in database systems are not sufficient.

The goal of retrieval is, given the description of the current problem, to retrieve the most similar (in the sense of being useful for solving the current problem, see Section 4.7.3) case or cases. In Figure 4.7 few of the possible ways to do retrieval are shown. The simplest form of retrieval may consist of doing similarity matching on all the cases in a cases-base and returning just one best matching case — this is called the nearest-neighbour retrieval (Figure 4.7a). However, this method may be too slow for large case-bases; therefore, often, a preselection of cases is done (Figure 4.7b). The preselection can be done using a simpler similarity measure, but, usually, it is done using the indexing structure of the case-base, for example, a decision tree. The problem with preselection is what to do when no match good enough has been found in the selected set of cases. Usually, preselection methods are only approximate, so there is a possibility that in the non-selected cases a still-better match can be found. Depending on the speed constraints, one may decide to search the rest of the case-base by widening the selection.

Apart from selection, one can use ranking (ordering) of cases to speed-up the retrieval. A simple ranking method is on the retrieval statistics for cases in a case-base. The cases which were retrieved most often and/or gave best matches can be considered as the prototypical cases and may be searched first. Another way of ranking is applicable to the clustered case-base organisation. One

may first calculate the matches of the current case with the cluster prototypes, and then search the clusters in the order determined by the values of the matches with the prototypes.

Retrieving only one case will work for many types of problems. However, there are problem domains where this is not the optimal solution. One reason for this might be the difficulty in defining a similarity measure which univocally determines the best match. Especially, when the cases are complex, determining the best match may depend on many, often subjective, factors. As already mentioned in Section 4.7.3, it is often easier to judge the optimality of solutions than the similarity of cases. Therefore, one may decide to retrieve several cases, use all of them to find the solutions, and then choose the best solution (Figure 4.7d). Another situation, where several cases may be retrieved, is when a solution is determined by a combination or interpolation between solutions from a number of cases (Figure 4.7c). Generally, the retrieval mechanism may be simpler and faster if a *larger number* of possibly similar cases are retrieved and their usefulness for problem solving is determined in the next stages, like adaptation and evaluation. This is because then the retrieval algorithm may be less accurate (less selective).

A possible way to speed-up the retrieval is to do the retrieval in parallel. This brings significant speed gains because matching a case does not require exchange of much information between the parallel running processes. Therefore, the speed-up should scale up well with the number of processing units. Implementation of parallel retrieval is simple for flat and clustered case-bases. Parallel searching of hierarchical case-bases is more difficult. Parallel retrieval can be implemented on high-end parallel hardware like Connection Machine [Kettler & Hendler, 1994]; however, also using a 2- or 4-CPU workstation brings significant speed gains, while only slightly increasing implementation costs and software complexity, see Section 7.5.6.

Normally, retrieval has to be preceded by a situation assessment — the situation has to be represented using the chosen case vocabulary. In some situations it maybe impossible or inefficient to fully describe the new problem in terms necessary for the retrieval before the retrieval is started. In some applications, this problem can be solved by incremental retrieval, where, given some initial description, a number of cases are retrieved, and afterwards the description may be adjusted or extended to retrieve more suitable cases. For incremental retrieval to be useful it has to be fast, guaranteeing short response times.

4.9 Adaptation

Generally, a retrieved case will not correspond to exactly the same problem as the problem for which the solution is being sought. This implies that the solution belonging to the retrieved case may be not optimal for the new problem and thus has to be adapted. During adaptation, depending on the differences between the retrieved and the current case, modifications to the solution are made to account for these differences. The possibilities for adaptation depend on the complexity of the solution. In case-based classification, where a solution consists of one of a few categories, usually no adaptation will be done.

Adaptation is usually done or guided by a set of rules. If just a few rules are present (e.g., to choose from a number of adaptation procedures) then they can be hard coded. More rules may require use of a rule-based subsystem for the adaptation. Sometimes, the knowledge of the problem domain may be good enough to let one design adaptation rules which could be able to generate the solution from scratch (this applies, for instance, to most planning problems). In such a situation the main gain from using CBR and using cases as basis for adaptation is the speed of finding solution and the reliability of these solutions.

Adaptation can be done iteratively. After each adaptation step an evaluation of the solution can be made. It may identify deficiencies in the solutions, suggesting further ways of adaptation. The iteration may go as far back as retrieval of further cases to serve as a starting point for a better solution. Iterative adaptation will be done especially in interactive CBR systems.

One can apply the case-based paradigm also to the adaptation and make use of cases of previous successful adaptation [Leake et al., 1996]. Such cases may define, for example, the successful application area of various adaptation methods.

Adaptation makes systems more complex but not necessarily significantly more powerful especially when a CBR system is used mainly to support a human in decision making. Adaptation may also reduce system reliability, this is especially an issue when mistakes made by the system are expensive. Therefore, in many situations no adaptation is done by the system and adaptation is done by the user instead. Mark et al. [1996] report that in a well-designed system the users do not perceive the need to do "manual" adaptation as something negative. On the other hand, mistaken automatic adaptations may significantly reduce users' confidence in the system.

4.9.1 Adaptation done by the system.

Adaptation done automatically by a CBR system can be divided into three categories [Smyth & Cunningham, 1993]:

- *Adaptation by substitution* (or by adjustment) — works for simple problems where the solution is in a form of a value or a collection of values with little interaction between them (e.g., tablet formulation systems described in [Craw et al., 1998]). This can include modifying a parameter in the appropriate direction, simple interpolation between several retrieved cases, or voting. In choosing how to adapt the parameters one can use simplified localised models of the problem domain. The possibilities of adaptation using this method are rather limited, resulting in low problem coverage of individual cases which, in turn, requires densely populated case-bases.
- *Transformational adaptation* — applies when there are interdependencies between the components of the solution. It involves structural changes to the solution and is applicable to, for example, planning problems. Transformation adaptation requires a deeper domain model but has more coverage than substitution.
- *Generative adaptation* — (also called derivational adaptation or reinstantiation) is done by reapplying the reasoning, previously used to solve the problem in the retrieved case, in the context of the new problem. Generally, it requires that, apart from the solution itself, also the steps which lead to that solution have to be stored with each case. This type of adaptation can be useful, for example, for design problems.

4.9.2 Adaptation done by the user

Most of the commercially used CBR systems have no provisions for automatic adaptation and all adaptation (if present) is done by the user. Even if no automatic adaptation is done, there are always some situations where the current problem is so similar to the retrieved case that the system can automatically use the old solution. In less clear situations the retrieved case can be used for providing a suggestion to the user. If the similarity can be (approximately) expressed using a single value, then such a system could use two thresholds to determine the proper action. Maes [1994] calls them a "do-it" and a "tell-me" thresholds. Given the possible range of similarity values and two thresholds, three situations are possible:

- similarity value above the "do-it" threshold — the system applies the solution automatically,
- similarity value above the "tell-me" threshold (and below "do-it") — the system gives a suggestion to the user who can accept it or adapt it if necessary,
- similarity value below "tell-me" threshold — the system informs the user that it does not know the solution.

In principle, one could use any solution below the "do-it" threshold as a suggestion, however, there exists a danger that some of these suggestions would be so incorrect that the confidence in the system would be reduced.

4.10 Solution evaluation & case storage

4.10.1 Evaluation before use

Usually, before a new solution can be applied to solve a problem, the correctness of the solution has to be evaluated. There are several possibilities how this can be done (these are not exclusive possibilities but all of them may be used by the same CBR system depending on the situation):

- *The solution is not evaluated but applied to the problem* — In principle this can be done if the cost of a failure is negligible, for instance, a help desk CBR application could suggest actions to solve some problem (this is a situation where some action may help but will not do any damage).
- *The solution is evaluated based on the similarity of the problems* — This is based on the assumption that solutions to very similar problems must also be similar. This will, for example, be used in situations when an automatic "do-it" threshold is used.
- *The solution is evaluated by the user* — This situation occurs often, for instance, in CBR for design where the user anyway has the ultimate say as far as the use of the design is concerned.
- *The solution is evaluated by the system* — This can, for example, be used when there is a model of the problem domain which is sufficient to evaluate the results of applying the solution. In a planning or design problem the satisfaction of all constraints may be checked automatically, for instance, cost of the plan execution can be estimated, etc. For constructions and structures one can for example check whether they will stand the expected work load.

4.10.2 Evaluation after use (feedback)

Feedback from the world is crucial to learning. Moreover, the evaluation of the correctness of the solution is, in general, much simpler after it has been applied to solve the problem. Even if the solution fails, this provides information useful for the system. In fact, failures may be more useful for learning than successes[1]. The failed case may suggest problem areas — the reasoner can use them to recognise its inadequacy in certain situations as far as some problem cases or some adaptation methods are concerned.

Learning from successes is simple to realise, mainly by storing the successful solutions as new cases. One can also store failed cases and use them to prevent the system from generating a wrong solution in the future. However, the system will learn most if an explanation for a failure has been provided. The first step to failure explanation is determining the reason for failure. A reason for failure may be inability to distinguish between cases — this may suggest use of new indexes or changes in the similarity measure. Another reason for failure is inability to predict that the suggested solution will fail — this suggests problems in the evaluation stage. A failure can be caused also by a fault in the adaptation rules.

One way to explain failure is to discover which parameters of the problem have caused a good match with a wrong case or what parameters are missing to prevent such matches in the future. After this is known, the weights of some features can be changed, new indexes can be added, wrong indexes can be inhibited, etc., see e.g., [Porter et al., 1990].

If a user is knowledgeable about the problem domain in which a CBR system works then learning can be based on the direct feedback from the user. If the user has not got enough knowledge about the problem domain (e.g., many help-desk systems) or there is no time to provide the failure explanations during the normal use of a system, then learning may be done off-line during the system maintenance. If the system learns on-line then some precautions have to be made to prevent the user from mistakenly entering inconsistent information.

1. There is a well known quote from Niels Bohr — "An expert is a man who has made all the mistakes which can be made in a very narrow field" (quoted in: Alan Mackay, *The Harvest of a Quiet Eye* (1977)).

4.10.3 Case storage

As already mentioned in the previous section, once the correctness of the solution has been evaluated, the new case can be stored in the case-base. Not every solved case has to be stored in the case-base. There is no use in storing many almost similar cases, for which the problem coverage largely overlaps. A simplest indication whether a case has to be stored may be the best match value or the amount of adaptation necessary to solve the problem represented by the case. One may also put limits on the size of case-base [Surma & Tyburcy, 1998] meaning that when a new case has to be added an old case may have to be removed.

Many systems do not automatically store new cases during use (on-line). The main reasons for this are: lack of time and insufficient domain knowledge of the users. The solved cases are often collected and entered into the case-base by more qualified personnel (this is discussed in Section 4.11.5 on case-base maintenance).

4.11 Developing CBR systems

4.11.1 When CBR should be used

One way to determine if CBR is a suitable methodology for a given problem is to consider if the advantages of CBR are important in the context of that problem. Of course, it is also necessary to consider if the disadvantages of CBR do not apply to the problem. A CBR system is a good candidate if positive answers can be given to the following questions:

- Is the problem already being solved in a CBR-like way? Do the experts base their solutions on the solutions to previous problems? This simplifies the conceptual design of the system.
- Are cases available? Are records being kept of the problems and solutions to these problems? Available records can often be converted into usable cases.
- Is the system mainly meant for assistance in problem solving? Will the system work in a cooperation with a user? This reduces the need for powerful adaptation and evaluation.
- Does the problem belong to a weak-theory domain? By basing solutions on what has already worked CBR may be able to find solutions even for such problems.
- Is the problem domain characterised by many exceptions from generally applicable rules? Cases provide a good way to handle exceptions where general knowledge about the problem is not sufficient.
- Is the problem to be solved computationally intensive? If previously solved problems are available they can serve as start-up solutions.
- Is the knowledge about the problem not complete or difficult to acquire? Does the problem have many yet unknown variations? The ability of CBR systems to learn can help to cope with such problems.
- Can it be expected that the system will require frequent maintenance (changes in the problem)? Will the maintenance be done by the users of the system? CBR can reduce the need for maintenance due to its ability to learn and adapt. If maintenance is still necessary then it is often easier than, for example, for rule-based systems.
- How important is a simple explanation of the solutions found by the system? By basing solutions on previous cases, reasons for new solutions are easy to explain.
- Can mistakes made by the system have serious legal consequences? Because of the reliance of the solutions on old cases, the reasons for failure can often be traced to mistakes made (by the user) in the past.
- Is the reliability crucial in the given problem domain? Are the costs of failure high? CBR systems can avoid mistakes by being able to recognise new, different problems for which they may be unable to provide good solution. The reliability can be increased by limiting the problem coverage of cases.

- Would the organisation which is to use the system profit from well structured and well accumulated knowledge? CBR systems are better for this purpose then database systems, because they provide better ways of retrieval. Also, when in use, a CBR system accumulates knowledge automatically.

Some problem characteristics which make CBR suitable because of being a lazy problem-solving method are (based on [Aha, 1997]):

- *Incremental learning* — Case-based approaches are suitable for problems requiring incremental (on-line) learning. The training speed is high.
- *Complex solution spaces* — Case-based approaches are suited for tasks where abstractions could yield over-generalisations. This is, for example, the case for highly disjunctive spaces, or concepts with exceptions.
- *Query-specific reasoning* — Lazy approaches can use the local information in response to queries. The eager abstraction approaches, which attempt to fit all cases simultaneously, may loose information that is crucial for specific queries. Lazy approaches can also be good at tolerating missing values, as the values needed to solve a specific problem are determined by each individual query.
- *Precedent explanations* — Because lazy methods do not discard the cases, it is easy to use them as precedent explanations. Abstract explanations can be generated if necessary.

Some of the characteristics of the problem domain which could speak against the use of the CBR are:

- There are no records of the problems and solutions available, or they are incomplete and would require significant effort to convert them into useful cases.
- The problem domain is dynamic in such a way that old solutions to the problems may no longer hold as good in the future. (The assumption of the regularity of the world does not hold.)
- The problem domain is characterised by presence of extraneous features (noise) which are irrelevant to finding good solutions but are difficult to separate from the problem description. This significantly increases the number of cases that the system will have to store and will reduce the system performance.
- There is a need for a fully automatic system, working without human assistance, but the domain theory is not fully circumscribed and one cannot guarantee full problem-domain coverage using the available cases.
- The system has to provide exact and optimal solutions to be acceptable. CBR systems may be incapable of it (providing only approximate solutions) if the available cases provide low problem space coverage and the adaptation knowledge is limited.
- Speed and/or memory is critical, and the CBR system would not perform satisfactorily if deployed on the intended hardware platform.

4.11.2 Design decisions

At the beginning of the development of a new CBR system several design decisions have to be made which will largely determine how the system will be constructed and how it will work. These decisions are based on, among other things, the amount of the expected user interaction with the system, the cost of failure of the solutions, amount of automatic adaptation, required speed of the system, etc.

User interaction. When developing a new CBR system one of the first questions one must answer is about the integration of the user with the system (this is usually determined by where and what for the system will be used). For example, the following situations can be distinguished:

- *The system will be fully automatic with no user interaction apart from off-line maintenance.* — Such a system can be successful if: there is full knowledge about the problem domain available, or it is acceptable that the system will solve only some of the problems. Full knowledge may imply availability of powerful adaptation rules or algorithms (or simplicity of the problem domain), or full problem domain coverage by the available cases. If one accepts a below 100% problem solution rate then one has to make sure that the system is capable of properly judging its area of competence. Simple systems, with a sub-100% solution ratio, may often be quite successful as they may contribute to workload reduction, by solving simple, mundane problems.
- *The system will work in interaction with a user (a "classic" CBR system).* — This means that the system will be able to solve a large percentage of the problems itself, but it will be always able to fall back on the user in order to do, for example, some adaptation if no similar enough case could be found, or the adaptation rules are not sufficient. The user may also do some evaluation and explanation and the system may learn on-line from the user.
- *The system will do only intelligent fast retrieval leaving the rest to the user.* — In such systems the emphasis is on the user interface (it has to be intuitive and flexible) and on the speed and flexibility of the retrieval.

One has to determine what type of interaction will take place and the way the user interaction will proceed. Will the user evaluate the solutions, do adaptation, explain failures? If so, then how will it be done? There is no point in implementing system features requiring user interaction which is unwanted by the user (e.g., costs too much time) or for which the user is not qualified. If a decision has been taken to have two groups of users – an ordinary user and a system manager with different responsibilities – different types of user interface may have to be implemented.

Because CBR systems typically require much user interaction, the design of the user interface itself is important. In general, user interface which lets the user to be in control of the way problem solving proceeds but does not force a certain work pattern on the user is preferable. Therefore, a Q&A type of interface, so typical of the old style expert systems should be avoided. A good user interface and properly designed interaction may be decisive about the eventual success of the system [Dearden, 1995; Göker et al., 1998].

Cost of failure. The other important question which has to be answered is the cost of failure. If it is low than one can allow the system to do several iterations, each time trying the solution out, to find the *right* solution. The system can also learn from the failures of the intermediate solutions. Low failure cost also means that the reliability of the system is not a critical factor.

Large cost of failure means that it is important to make sure that the solution is correct before it is applied to solve the problem. The evaluation of the correctness will be often done by the user or based on detailed models of the problem domain if such are available. One will strive to have as many cases as possible to obtain good problem-domain coverage without need for much adaptation, and to simplify evaluation of the correctness of solutions. Because one will explicitly avoid any failures there will be also little possibility to learn from them.

Other design decisions. A development of a CBR system usually requires also the following design decisions:

- Determine the content of the case. This will usually imply the choice of the vocabulary to describe the relevant parameters, features of the problem situation.
- Determine the similarity measure. How the similarity between the cases will be calculated? What is the importance of the various features of the problem? One has to define the similarity such that is based on the problem solving capability of the matching cases, i.e., given a new problem such cases have to be retrieved which can be used to solve that problem.
- Decide the balance between the number of cases necessary and the power of the adaptation rules. Both have to lead to as full as possible problem-domain coverage. The choice can be

determined by factors like: is this a weak-theory domain problem, are there cases available, what is the cost of failure (in general cases are more reliable than theory, moreover, a failure caused by wrong cases can be blamed on the user, a failure caused by faulty rules will be blamed on the system designer), etc.

- Determine the required speed of the system and the ways of achieving it. Can it be achieved by just using fast hardware? Does the case-base require special design to achieve the required retrieval speed? What type of case-base organisation should be chosen?

One cannot predict everything, but the design of a CBR system should take the occurrence of the unexpected into account, especially, if not all its actions are supervised by the user. This involves issues like, for example, situation assessment and case representation. The system should be able to distinguish between problem situations for which it can find solutions and problem situations for which it does not know a solution. This may be problematic if, for example, the problem representation (vocabulary) is incapable of capturing relevant problem features, or if the similarity measure ignores differences which are after all significant. Serious errors can be made if the decisions on case representation are based only on the (limited number of) available cases, for instance, using statistical analysis, without a proper understanding of the problem domain.

4.11.3 Knowledge engineering

Section 4.3 has already mentioned simplified knowledge acquisition as one of the advantages of CBR. How the knowledge acquisition in a given CBR project will proceed is largely determined by the chosen knowledge distribution over the various knowledge containers as discussed in Section 4.3. Distribution of the knowledge over the (sub-)containers provides a structure for the knowledge, which should simplify knowledge acquisition.

Generally, CBR systems are more reliable and easier to build and maintain when the number of cases is large. This way the whole problem space is covered by many cases each having only a small problem coverage. This, in turn, means that the adaptation rules can be simpler. The less adaptation is done by the system the less deep the required knowledge can be and thus easier to formalise. Another approach is to use a smaller number of carefully selected "golden" cases which have a wider problem coverage because the adaptation knowledge is more extensive.

Acquisition of the knowledge to be used in the adaptation stage will be similar to the normal knowledge acquisition for rule-based systems, however, usually, knowledge about adaptation of a solution is much easier to acquire than knowledge about the derivation of the solution. Moreover, knowledge about adaptation can be better structured based on the types of problems (cases) it applies to. Models of the problem domain can be simpler and relatively easy to acquire if they are not all-encompassing but apply only to groups of similar cases.

The knowledge acquisition for a CBR system can be done iteratively. The initially acquired knowledge can be implemented in a prototype system. One can then test it on the available data, record the problems (failures), and discuss them with an expert and based on his explanations, make necessary changes to the system. This makes knowledge acquisition well directed, specific, and efficient.

Acquiring cases. What constitutes the core of a CBR system are cases. Therefore much of the "classical" knowledge engineering is replaced by "case engineering" [Aha, 1997]. In order to obtain the cases one has to have the records of previously solved problems. If such are not present or not available then a collection of problems has to be established and these have to be solved by an expert.

In practice, records of previously solved problems are often not kept or they are incomplete [Mark et al., 1996; Göker et al., 1998]. Typical examples of incomplete records are:

- Records were to be maintained by the users and recording of information was either voluntary or not strictly reinforced, therefore, the records have significant gaps.

- There may have been records kept of "problems" but not of solutions, for example, in NDT the raw data may be available but it is not classified, or when a certain situation is being monitored the problem and the results of applying a solution may be recorded, but not the solution itself.
- Only the "important" problem/solution pairs may be recorded; this gives us an incompletely and unevenly covered problem space.
- A complete record of problems and solutions may be available but there is no record of how the solutions have been achieved (for example, for design problems). This can be a problem for situations where the problem space is sparsely covered and there exists need for significant solution adaptation (e.g., generative adaptation).

When good, representative, and well-documented cases are available, then there is much less need for knowledge acquisition from the experts.

4.11.4 System speed

Kolodner [1993] names the problem of guaranteeing acceptable retrieval speeds a "major technological issue." Since 1993, the speed of CPUs has increased more than 20 fold and the memory is about 50 times cheaper. Still, achieving acceptable speed for systems working with large case-bases and complex cases has remained an issue. Especially for multimedia type cases, guaranteeing fast retrieval times is a problem.

As usual in software engineering, improving processing speed can be achieved by faster hardware or by clever algorithms. In CBR a "clever algorithm" usually means an efficient case-base organisation and a corresponding retrieval scheme (including, possibly, parallel retrieval). If a system is to be deployed on a small number of computer systems then it makes sense to invest in faster hardware instead of optimising the case-base organisation, as this may be economically a better choice. Getting the most of the case-base organisation may mean that no use can be made of standard CBR tools and the system has to be coded from scratch, for example, in C++.

4.11.5 Maintenance

In Section 4.4 the ease of maintenance has been mentioned as one of the advantages of CBR. In some situations CBR systems require no maintenance by the users. This usually occurs when the problem domain is stable and any changes can be compensated by automatic adaptation (acquiring new cases). However, usually, explicit maintenance is necessary, especially in problem domains where cases which lead to good solutions in the past may suggest wrong solutions at present (e.g., computer configuration). In principle all types of knowledge as listed in Section 4.3 may need maintenance. However, most of the maintenance is done on the case-base. Other types of maintenance, like changing the adaptation rules, adjusting local functional models, etc. are much less frequent.

Leake and Wilson [1998] provide the following definition of case-base maintenance: *Case-base maintenance implements policies for revising the organisation or contents (representation, domain content, accounting information, or implementation) of the case-base in order to facilitate future reasoning for a particular set of objectives.* Leake and Wilson also propose a framework for describing methods of case-base maintenance. The framework describes maintenance policies according to the following parameters:

- *Data collection* — has as goal gathering information about the cases, case-base (part or whole), or the whole system. This information is then used to determine what maintenance actions should be taken.
- *Triggering* — Based on the collected data, maintenance actions can be triggered. The timing can be: periodic, conditional, or ad hoc. In conditional triggering, the conditions can be: space-based (related to the case-base size), time-based, or result-based (based on evaluation of performance).

- *Operation types* — determine what actual maintenance actions will be performed. They can be divided based on the target of the operations (indexing, domain contents, accounting, or maintenance policies themselves) and on the revision level (implementation, representation, or knowledge level).
- *Execution* — does not have to follow the triggering but may be deferred until a more appropriate time. It is defined by the scope of the maintenance operations: broad (concerning whole case-base) or narrow (concerning one or a few cases).

As one can notice, according to this definition and framework description also on-line adding of cases to the case-base is considered as case-base maintenance. A more typical, explicit case-base maintenance usually takes one of the two forms:

- *Off-line addition of new cases to the case-base* — This applies, for example, to CBR systems for help desks. One has to analyse the problems and the solutions, verify their correctness (users of the system may make mistakes), estimate their usefulness, and perform necessary corrections before they can be added to the case-base.
- *On-line or off-line deletion of cases* — This may be necessary to maintain the required response times. A number of strategies have been developed for choosing which cases are to be deleted. A decision about deletion may be made simply based on the retrieval statistics (accounting information). One may also base the decision on, for instance, an estimate of the competence of the cases concerned [Smyth & McKenna, 1998], or the location of cases with respect to their neighbours [Surma & Tyburcy, 1998].
- *Analysing a case-base* — This will usually be done after some time of use of a system which allows on-line case addition by the end-users. One can remove unwanted cases (e.g., these, which over-represent some problem types or which represent problem rarely likely to occur); the indexing structure can be adapted (e.g., a decision-tree case-base structure can be reorganised); etc.

As far as the maintenance is concerned one can make a distinction between a number of user groups each having different competencies and different responsibilities. One possible distinction is into three groups:

- *Direct users of a CBR system* — can be responsible for evaluating, describing, and adding the cases.
- *Case-base managers* — designated by the organisation using a CBR system. They may do the case entering, analysis of the case-base, removal of cases. They can also prepare new problem-specific case-bases from the set of cases available in the organisation.
- *Developer of the system* — may have to be involved if, for example, the case-base organisation, the indexing, or the adaptation rules or procedures have to be changed.

For example, Göker et al. [1998] distinguish three user types for their HOMER system: Help-Desk Operator, Case-Base Editor, and System Administrator. Help-Desk Operator uses the system to solve the problems, new solved problems are recorded as new cases and stored in a case-buffer. Case-Base Editor checks the cases in the case-buffer and transfers the relevant ones to the main case-base. He also checks the redundancy and the consistency of the case-base. The System Administrator maintains the underlying domain and case model and administers the user rights.

Maintenance may require use of special tools, different from the CBR system used for on-line problem solving. Some facilities such tools may provide are:

- Fast and convenient entering of specification for the new cases.
- Filtering, selection, statistical analysis, displaying of the cases contained in the case-base(s).
- Interface to an external database which may contain cases, original data which was used to construct the cases, product descriptions, etc.

Whenever a number of case-bases are used (e.g., problem specific case-bases) an issue of case-base management arises. One has to have tools and methods for preparing the case-bases to be used by

a CBR system from the available pool of cases, to merge the case-bases, to analyse their consistencies (if cases can be added on-line), etc. Also, administering of case-bases is important. For example, in situations where some solutions failed and the failure was established only after a period of time has passed, one must to be able to trace which particular configuration has led to failure.

4.11.6 Analysing CBR systems

In [Althoff, 1995] a number of criteria are presented which can be useful for evaluating existing CBR systems and tools, and also useful for making decisions when building a new CBR system. The criteria are mainly qualitative, as quantitative criteria are too much dependent on specific measurement procedures (therefore difficult to be used for comparisons) and measuring only isolated aspects of the systems. The criteria are organised into two groups:

- *Domain independent qualitative criteria* — describe capabilities of CBR systems and can be used for their comparison:
  - Technically oriented criteria:
    - (methods for) case and knowledge representation, e.g., flat vs. structured case representation,
    - organisation of the case library, e.g., type of indexing used,
    - similarity measure(s), e.g., ways of combining local similarities,
    - handling of noisy and incomplete data (cases with unknown attribute values),
    - performance (speed, memory requirements).
  - Criteria related to the development process and system use:
    - acquisition and maintenance of knowledge and data (modelling support, facilities for entering cases),
    - explanation ability,
    - validating and testing of the application system, e.g., facilities for running test scripts and output of result statistics,
    - user acceptance.
- *Domain and application task dependent qualitative criteria* — describe the requirements that arise from a certain domain and application task, they have to be satisfied if an application problem is to be solved:
  - General criteria, for example:
    - Size: number of cases, classes, attributes and attribute values,
    - Complexity, e.g., number of different relationships between case elements,
    - Theory strength: certainty of relationships,
    - Openness: dependency on (assumptions about) external factors,
    - Change over time: static vs. dynamic.
  - Concept structure, for example:
    - Neighbourhood notion available?
    - Single or multiple reasoning goals.
    - Test selection necessary, i.e., acquisition of additional symptom values?
  - Integration of reasoning strategies:
    - alternating — choice of the most appropriate strategy depending on a problem to be solved,
    - parallel — if results are different arbitration is necessary,
    - interleaved — one strategy supplement another,
    - seamless — e.g., combination of induction and CBR.

These types of criteria have, for instance, been used by Althoff et al. [1995] for the purpose of evaluating a number of case-based reasoning tools, see Section 4.12.3.

4.12 CBR in practice

4.12.1 Application domains

There exist a large number of commercially successful and/or academically interesting CBR systems. The typical domains for which CBR systems have been developed are listed below together with some system examples. (The list is based on [Watson & Marir, 1994], descriptions of most of the systems can be found in [Riesbeck & Schank, 1989; Kolodner, 1993], unless other references are given.) There is a large overlap between the domains, so often a system can be considered as belonging to more than one domain, for instance, classification systems can be considered as either interpretation or as diagnosis systems depending how the problem is formulated.

Interpretation / Legal reasoning. Interpretation is a process of evaluation of situations/problems is some context. Commonly, interpretation is based on previous experience. Interpretative CBR is typically used in legal reasoning. This is understandable because the practice of law is largely based on precedents (previous law cases). Examples of academic systems are: JUDGE (1986) – for criminal sentencing, HYPO (1988) – an interpretive reasoner for patent law, GREBE (1991) – damage claims, KICS [Yang et al., 1994] – building regulations.

Diagnosis / Explanation / Classification. In diagnosis problems, usually, an explanation has to be found for a number of symptoms. If the number of possible explanations is small then diagnosis can be viewed as classification. Explanations may be difficult to generate, but easy to verify. Medical diagnosis was the domain of the first expert systems (e.g., MYCIN). However, usually, diagnosis is made based not on a number of rules but based on similarity to old cases. No wonder that also CBR systems have been used for medical diagnosis. For example PROTOS [Porter et al., 1990] for case-based classification of hearing disorders and CASEY (1989) for diagnosis of heart problems. LISSA and the CBR system for URS described in Chapters 6 & 7 are examples of CBR used for classification.

Design. In design, problems are described in terms of constraints that have to be satisfied. The solutions to design problems are usually compound (have structure). Typically, there are many possible solutions satisfying the constraints and there can be additional criteria used to choose one of the designs. Sometimes, the problem may be overconstrained so compromises have to be made or the problem has to be respecified. Design problems are mostly quite complex and solving them often requires decomposition, as the individual full cases will rarely match well. Examples of CBR systems for design are: JULIA (1992) for meal planning, ARCHIE (1992) for conceptual design in architecture, Déjà Vu [Smyth, 1996] for producing control software, EADOCS [Netten, 1997] for design of composite sandwich panels for aircraft, and CBR-TFS [Craw et al., 1998] for tablet formulation. For complex design problems even retrieval-only systems can still be useful.

Planning. Planning is similar to design, but deals with arranging a sequence of actions in time. There is a certain goal that has to be achieved, a set of constraints that have to be met, and some fitness criterion that has to maximised (one usually wants the plan to be (near) optimal). Most planning problems can be solved by brute search, but are usually computation intensive. Examples of CBR planning systems are: CHEF (1989) which prepares recipes viewed as plans, BOLERO [Lopez & Plaza, 1993] which builds diagnostic plans for medical patients, and TOLTEC [Tsatsoulis & Kashyap, 1993] which prepares manufacturing process plans based on the part specifications.

Help desks / customer support (advising). This is typically a diagnosis type of problem, but is distinguished by emphasis on case retrieval. Customer-support help desks deal with all kinds of software and hardware problems and faults. Usually, it is not economically justifiable to have highly qualified personnel man help desks. Therefore, help desks are usually served by a less qualified personnel using a set of tools to help them answer incoming problem calls. Usually, it is difficult to

model all the possible interactions between components of the system to be diagnosed and its environment; therefore, a model-based reasoner would have a limited success for such applications. CBR can be useful without the need to model the diagnosed domain. Many CBR tools (Section 4.12.3) are mainly for help-desk CBR applications.

An example of a CBR system in this domain is HOMER [Göker et al., 1998] for CAD/CAM software/hardware help desk. Even more commercially attractive are CBR systems available via manufacturer Internet sites, where users themselves can find answers to the problems they have with software or hardware products. An example of such a system is Gizmotapper deployed by Brøderbund (the maker of "Myst" and "Riven" games) at http://seudo.broder.com/.

4.12.2 Commercial applications

This section presents several commercially successful applications of CBR (based on [Watson & Marir, 1994a]).

Lockheed's CLAVIER. CLAVIER [Kolodner, 1993; Mark et al., 1996] is probably the first commercially used CBR system. It has been designed to automate determination of optimal layouts of composite aeroplane parts to be heated in an autoclave (a large convection heater). The heating characteristics of the whole process are not fully understood. Finding an optimal layout is not easy because of the wide variety of parts with various heating characteristics and shapes. The choice of parts forming a layout is also determined by the priority of the incoming parts. Before the CBR system was implemented, when a new set of parts was to be put in the autoclave, an operator tried to find previous similar layouts (stored as drawings) and then made necessary changes to them. Especially making the necessary adaptations required considerable expertise. Because this procedure closely resembled the CBR paradigm, when Lockheed decided to automate the process, CBR was chosen. One of the perceived advantages was the ability of the CBR system to learn. CBR was not the only considered alternative. Lockheed also considered using rule-based technology, thermodynamic modelling, and inductive learning as candidates, but none of them proved practical.

The CLAVIER project began in 1989. A number of problems were encountered during the project. For example, it has turned out that the records kept were inadequate to determine validity of the layouts. Therefore, though at first it seemed that the cases were already there, a considerable effort was needed to assemble an initial collection of good cases. Though, the initial intention was to have the system do automatic adaptation, it proved to be too difficult and it was decided to let an operator do the adaptation of the retrieved cases; the system then can do the validation of the new layout. Validation is partly done by checking if the layout does not match any previous invalid layouts.

It was expected that the case-base would stop growing (at about 300 cases) after a period of about two years (two to three loads are cured each day). However, after putting the system into use (September 1990) the case-base kept on growing and in 1995 contained more than 600 cases. This has been explained by the dynamic nature of the manufacturing process. Also, it has turned out that the classification of the load into only two categories, valid/invalid, is not fully sufficient to reflect the evaluation of the quality of the layouts.

CLAVIER ensures that high quality layouts are used even when the experienced operators are not available. It maintains also the consistency of layouts. Apart of solving the problem, CLAVIER also serves as a collective (corporate) memory, allowing for easy transfer of knowledge between autoclave operators. Since CLAVIER became operational the incompatible layouts have been virtually eliminated, saving thousands of dollars each month.

British Airways' CaseLine. CaseLine assists with fault diagnosis and repair of Boeing 747 between arrival and departure. When a plane is on the ground, any reports of detected or suspected faults are analysed to determine the best actions to find the cause and to repair it. CaseLine maintains a case-base of previous failure instances and details of recovery actions. This case-base can be

searched using either a fault code or description. The retrieved case helps identify the procedures with best chance of success.

According to British Airways the benefits of CBR are: the system is intuitive to developer and users, it complements human problem solving, and it retains the context of the problem situation. The last point is especially important because of the legal reasons. Because the cases are contextualised it is easy to explain the reasons for a particular decision. This is especially important in the case of failure of a procedure suggested by the system.

Legal & General's SWIFT. Legal & General is a UK based financial company. They needed a system that could handle ordering of PC products and upgrades. Because of the volatility of the PC market, the system should be easy to maintain. Because of this constraint CBR was chosen as the most suitable knowledge-based technology.

The system was implemented partly using the CBR-Express/CasePoint combination (see Section 4.12.3). SWIFT analyses requests from employees, retrieves several matching cases, determines which is the best one in a number of aspects, and offers a solution. Unresolved cases are analysed periodically to determine whether either the existing cases are to be adapted or new cases have to be added.

The most important thing that this system demonstrates is the short time needed for development and that a CBR system can be maintained by people who are not programmers (developers) of the system.

CBR-URS (ANDES project). Within the ANDES project two applications have been developed for NDT data interpretation. These are described in detail in Chapters 6 & 7. The system developed for the interpretation of data from ultrasonic rail-inspection system (URS) is currently being developed into an application that can be used on the inspection train.

4.12.3 Examples of existing CBR tools

There are a large number of software tools available for building CBR systems. In the following, several most popular CBR development tools are presented, based mainly on [Watson & Marir, 1994]. A presentation and comparison of these and other CBR tools can also be found in [Watson, 1997]. Althoff et al. [1995] present a detailed evaluation of five tools: CBR-EXPRESS, ESTEEM, KATE, REMIND, and S^3-CASE.

- *CBR-EXPRESS* — by Inference Corp. (www.inference.com) is tailored for the help-desk market. CBR-EXPRESS has a simple case structure and uses nearest-neighbour matching in case retrieval. A key feature of CBR-EXPRESS is that it can handle freeform text. It does not use a fixed list of question and answers. The text describing a problem is analysed lexically (not semantically) and cases with most similar problem descriptions are retrieved (the matching algorithm is fast so that it can handle large databases). Unresolved cases (for which there was no match) are later processed by a case-base administrator.
- ART**Enterprise* — by Brightware (www.brightware.com – a subsidiary of Inference Corp.). This tool derived from ART (Advanced Reasoning Tool – an expert system shell/inference engine) includes the CBR-EXPRESS functionality adding to it a variety of representation possibilities like objects and rules. It comes with a GUI builder and facilities for DBMS access. The combination of various techniques is interesting but the CBR module itself is not very powerful.
- *The Easy Reasoner for Eclipse* — from Haley Enterprises (www.haley.com). It is a C library for Eclipse (an object oriented expert system shell derived from ART) that provides CBR functionality. It features nearest-neighbour and inductive (decision tree) retrieval. Retrieved cases can be processed by Eclipse. A usual range of variable types can be handled, and the text handling possibilities are the same as in the two systems from Inference Corp.

- *REMIND* — by Cognitive Systems Inc. is (or rather was as Cognitive Systems ceased trading in 1996) one of the more flexible CBR tools. It offers template (queries), weighted nearest-neighbour, inductive, and knowledge-guided inductive retrieval. Inductive retrieval involves building a decision tree that indexes the cases. Qualitative model can be built which, for example, defines which features (concepts) depend on other features. This model can be used to guide a tree induction. Case adaptation is also possible by adjusting values in a solution based on the differences between cases. Its text handling capabilities are less powerful than these of the above described products. It does not offer freeform text input.
- *RECALL* — by ISoft (www.alice-soft.com) has a full featured object oriented language for case description. It has hierarchical indexes for case-base organisation and retrieval. It also uses a variant of nearest-neighbour algorithm. Similarity functions can take into account structural differences between cases. It has a standard adaptation mechanism based on a voting and a possibility for user defined adaptation rules. It has a graphical UI for definition and modification of the cases, case-base, adaptation, etc. The system is extendable — it can be easily interfaced to external applications (it is available as a C++ library) and it can interpret Tcl code [Ousterhout, 1994].
- *CBR-WORKS* — (previously S^3-CASE) by tecInno (www.tecinno.de) is a client/server system. The CBR-WORKS server (written in SMALLTALK, which makes it largely platform independent) contains a case-base and a domain model and can respond to queries from a client. It can also access external databases. The client is usually written (e.g., in Java, HTML, or CGI) specifically for the intended applications, so that its user interface can be developed to suit the application domain. Routines for communication with the server are provided with the CBR-WORKS development environment. Queries can also be done using platform-independent CASUEL or CQL (case query language). CBR-WORKS allows building complex object-oriented domain models which are used to structure the cases. The domain model can be used directly by both the client and the server. An example of an application built using CBR-WORKS is HOMER described in [Göker et al., 1998].

Most of the tools have been designed for help-desk type of problems and thus are suited mainly to processing cases described in text form or as attribute/value pairs (flat representations), which makes them unsuitable for use with complex-structured cases like images, design descriptions, etc. Moreover, the flexibility with which the similarity measure can be specified is often limited to a number of predefined choices. Out of the five tools evaluated in [Althoff et al., 1995] only two, KATE and S^3-CASE, are capable of representing complex-structured cases, and only S^3-CASE allows user-programmable similarity measures.

Generally, tools which provide interfaces to external routines, like RECALL's ability to execute Tcl code (e.g., for user-defined adaptation), and S^3-CASE's ability to execute SMALLTALK code, are the most flexible ones. Still, many CBR systems, especially those for complex and novel application problems, are often built without a use of special CBR tools, for example, EADOCS [Netten, 1997] or the two systems described in Chapters 6 & 7.

4.12.4 Future of CBR

Considering the large number of advantages as listed in section 4.5 CBR has certainly a bright future. Whereas currently most CBR systems are stand-alone systems, in the future one can expect emergence of many applications where CBR is integrated into broader systems. Already it can be noticed that many CBR tools, like CBR-WORKS for example, provide interfaces that can be accessed from other applications.

Some areas where more research is needed are:

- case representation methods, especially for multimedia type data like images, sounds (or signals in general),
- efficient handling of huge case-bases,
- adaptation, largely neglected in commercial systems, but crucial to planning and design domains,
- easing knowledge acquisition (e.g., by integration with other learning methods) and verification, simplifying authoring of cases, extracting cases from unstructured data (e.g., text),
- maintenance of case-bases.

Though it has so many advantages, CBR is not the silver bullet[1] of AI — it does not solve all the problems and is not always the best methodology. For some problems other Artificial Intelligence techniques can be more suitable. Often, the best results can be obtained by hybrid systems using a combination of techniques. That is why many commercial tools offer CBR capabilities in combination with inductive learning, rule-based reasoning, and other techniques.

4.13 References

Aamodt, A. (1993) "A Case-Based Answer to Some Problems of Knowledge-Based Systems", *Scandinavian Conference on Artificial Intelligence*, E. Sandewall and C.G. Jansson (eds.), IOS Press, pp. 168–182.

Aamodt, A. and Plaza, E. (1994) "Case-based reasoning: Foundational issues, methodological variations, and system approaches", *AICOM*, Vol. 7, No. 1, March, pp. 39–59.

Aha, D.W. (1997) *The Omnipresence of Case-Based Reasoning in Science and Application*, Washington, DC: Naval Research Laboratory, Navy Center for Applied Research in Artificial Intelligence, (Technical Report AIC-98-002).

Aha, D.W. (1998) "Reasoning and Learning: Exploiting the Lazy-Eager Dimension", Invited talk EWCBR-98, Dublin.

Aha, D.W., Kibler, D., and Albert, M.K. (1991) "Instance-Based Learning Algorithms", Machine Learning, Vol. 6, pp. 37–66.

Althoff, K.-D. (1995) "Evaluating Case-Based Reasoning Systems", *Proc. Workshop on Case-Based Reasoning: A New Force In Advanced Systems Development*, Unicom Seminars Ltd., pp. 48–61.

Althoff, K.-D., Wess, S., Bergmann, R., Maurer, F., Manago, M., Auriol, E., Conruyt, N., Traphöner, R., Bräuer, M., and Dittrich, S. (1994). "Induction and Case-Based Reasoning for Classification Tasks", H.H. Bock, W. Lenski, and M.M. Richter (eds.), *Information Systems and Data Analysis, Prospects-Foundations-Applications, Proc. 17th Annual Conference of the GfKl, University of Kaiserslautern*, Springer-Verlag, Berlin-Heidelberg, pp. 3–16.

Althoff, K.-D., Auriol, E., Barletta, R., and Manago, M. (1995) *A Review of Industrial Case-Based Reasoning Tools*, AI Intelligence, Oxford, UK.

Bergmann, R. and Stahl, A. (1998) "Similarity Measures for Object-Oriented Case Representations", *Advances in Case-Based Reasoning: EWCBR-98 Proceedings*, B. Smyth and P. Cunningham (eds.), Springer-Verlag, pp. 25–36.

1. F.P. Brooks used the term "silver bullet" to describe the perfect (magical) solution to the problems of software development ("No Silver Bullet", *IEEE Computer*, Vol. 20, No. 4, April 1987, pp. 10–19).
— *"Of all the monsters that fill the nightmares of our folklore, none terrify more than werewolves, because they transform unexpectedly from the familiar into horrors. For these, one seeks bullets of silver that can magically lay them to rest. The familiar software project, at least as seen by the nontechnical manager, has something of this character; it is usually innocent and straightforward, but is capable of becoming a monster of missed schedules, blown budgets, and flawed products. So we hear desperate cries for a silver bullet – something to make software costs drop as rapidly as computer hardware costs do."*

Bonzano, A. (1998) *ISAC: a Case-Based Reasoning System for Aircraft Conflict Resolution*, Ph.D. Thesis, University of Dublin, Trinity College, Ireland.

CASUEL (1994) *CASUEL: A Common Case Representation Language*, http://wwwagr.informatik.uni-kl.de/~bergmann/casuel/

Craw, S. (1998) *Easing Knowledge Acquisition for Case-Based Design*, http://www.scms.rgu.ac.uk/research/kbs/kacbd/kacbd.html

Craw, S., Wiratunga, N., and Rowe, R. (1998) "Case-Based Design for Tablet Formulation", *Advances in Case-Based Reasoning: EWCBR-98 Proceedings*, B. Smyth and P. Cunningham (eds.), Springer-Verlag, pp. 358–369.

Dearden, A.M. (1995) "Improving Interfaces to Interactive Case Memories", *Progress in Case-Based Reasoning*, I. Watson (ed.), LNAI 1020, Springer-Verlag, Berlin, pp. 73–84.

Fisher, D.H. and Schlimmer, J.C. (1988) *Models of Incremental Concept Learning*, Vanderbilt University Technical Report CS-88-05 (http://cswww.vuse.vanderbilt.edu/~dfisher/tech-reports.html).

Göker, M., Roth-Berghofer, T., Bergmann, R., Pantleon, T, Traphöner, R., Wess, S., and Wilke, W. (1998) "The Development of HOMER – A Case-Based CAD/CAM Help-Desk Support Tool", *Advances in Case-Based Reasoning: EWCBR-98 Proceedings*, B. Smyth and P. Cunningham (eds.), Springer-Verlag, pp. 346–357.

Hanney, K. and Keane, M.T. (1997) "The Adaptation Knowledge Bottleneck: How to Ease it by Learning from Cases", *Case-Based Reasoning Research and Development (ICCBR-97)*, D.B. Leake and E. Plaza (eds.), Springer-Verlag, Berlin, pp. 359–370.

Kettler, B.P. and Hendler, J.A. (1994) "Massively Parallel Support for Case-based Planning", *IEEE Expert*, February, pp. 8–14.

Kolodner, J. (1993) *Case-Based Reasoning*, Morgan Kauffman Publishers, Inc., San Mateo, CA.

Kolodner, J. (1996) "Making the Implicit Explicit: Clarifying the Principles of Case-Based Reasoning", *Case-Based Reasoning: Experiences, Lessons & Future Directions*, D.B. Leake (ed.), AAAI Press, Menlo Park, CA, pp. 349–370.

Leake, D.B. (1996) "CBR in Context: The Present and Future", *Case-Based Reasoning: Experiences. Lessons & Future Directions*, D.B. Leake (ed.), AAAI Press, Menlo Park, CA, pp. 3–30.

Leake, D.B., Kinley, A., and Wilson, D. (1996) "Acquiring Case Adaptation Knowledge: A Hybrid Approach", *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, AAAI Press, Menlo Park, CA, pp. 96–104.

Leake, D.B. and Wilson, D.C. (1998) "Categorizing Case-Base Maintenance: Dimensions and Directions", *Advances in Case-Based Reasoning: EWCBR-98 Proceedings*, B. Smyth and P. Cunningham (eds.), Springer-Verlag, pp. 198–207.

Lopez, B. and Plaza, E. (1993) "Case-based planning for medical diagnosis", *Methodologies for Intelligent Systems, 7th International Symposium, ISMIS-93 – LNAI 689*, Springer-Verlag, pp. 96–105.

Maes, P. (1994) "Agents that Reduce Work and Information Overload", *Communications of the ACM*, July, Vol. 37, No. 7, pp. 31–40.

Malek, M. (1995) "A Connectionist Indexing Approach for CBR Systems", *Case-Based Reasoning Research and Development (ICCBR-95)*, M. Veloso and A. Aamodt (eds.), Springer-Verlag, Berlin, pp. 520–527.

Malek, M., Toitgans, M.-P., Wybo, J.-L., and Vincent, M. (1998) "An Operator Support System Based on Case-Based Reasoning for the Plastic Moulding Injection Process", *Advances in Case-Based Reasoning: EWCBR-98 Proceedings*, B. Smyth and P. Cunningham (eds.), Springer-Verlag, pp. 402–413.

Mark, W., Simoudis, E., and Hinkle, D. (1996) "Case-Based Reasoning: Expectations and Results", *Case-Based Reasoning: Experiences. Lessons & Future Directions*, D.B. Leake (ed.), AAAI Press, Menlo Park, CA, pp. 269–294.

Netten, B.D. (1997) *Knowledge Based Conceptual Design: An Application to Fibre Reinforced Composite Sandwich Panels*, Ph.D. Thesis, Delft University of Technology, The Netherlands.

Ousterhout, J.K. (1994) *Tcl and the Tk Toolkit*, Addison-Wesley Pub. Co.

Perner, P. (1998) "Different Learning Strategies in a Case-Based Reasoning System for Image Interpretation", *Advances in Case-Based Reasoning: EWCBR-98 Proceedings*, B. Smyth and P. Cunningham (eds.), Springer-Verlag, pp. 251–261.

Porter, B.W., Bareiss, R., and Holte, R.C. (1990) "Concept Learning and Heuristic Classification in Weak-Theory Domains", *Artificial Intelligence Journal*, Vol. 45, No. 1–2, pp. 229–264.

Quinlan, J.R. (1993) *C4.5: Programs for Machine Learning*, Morgan Kaufmann Publishers, Inc., San Mateo, CA.

Riesbeck, C.K. and Schank, R.C. (1989) *Inside Case-Based Reasoning*, Hillsdale, Erlbaum.

Richter, M.M. (1995) *The knowledge contained in similarity measures.* Invited talk at the ICCBR-95 (http://wwwagr.informatik.unikl.de/~lsa/CBR/Richtericcbr9595remarks.html).

Schank, R.C. (1982) *Dynamic Memory: A theory of reminding and learning in computers and people*, Cambridge University Press.

Schank, R.C. and Abelson, R.P. (1977) *Scripts, Plans, Goals and Understanding*, Erlbaum.

Schmidt, R. and Gierl, L. (1998) "Experiences with Prototype Designs and Retrieval Methods in Medical Case-Based Reasoning Systems", *Advances in Case-Based Reasoning: EWCBR-98 Proceedings*, B. Smyth and P. Cunningham (eds.), Springer-Verlag, pp. 370–381.

Smyth, B. (1996) *Case-Based Design*, Ph.D. Thesis, Department of Computer Science, Trinity College Dublin, Ireland, (http://compsci.ucd.ie/staff/bsmyth/).

Smyth, B. and Cunningham, P. (1993) "Complexity of Adaptation in Real-World Case-Based Reasoning Systems", *Artificial Intelligence & Cognitive Science VI*, Belfast, Queens University Press.

Smyth, B. and McKenna, E. (1998), "Modelling the Competence of Case-Bases", *Advances in Case-Based Reasoning: EWCBR-98 Proceedings*, B. Smyth and P. Cunningham (eds.), Springer-Verlag, pp. 208–220.

Thompson, V. (1997) "Corporate Memories", *Byte*, September, pp.(32IS) 7–12 (http://www.byte.com/art/9709/sect17/art1.htm).

Tsatsoulis, C. and Kashyap, R.L. (1993) "Case-Based Reasoning and Learning in Manufacturing with the TOLTEC Planner", *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 23, No. 4, July/August, pp. 1010–1023.

Turing, A.M. (1950) "Computing Machinery and Intelligence", *Mind*, Vol. 59, pp. 433–460.

Watson, I. (1997) *Applying Case-Based Reasoning: Techniques for Enterprise Systems*, Morgan Kaufmann Publishers.

Watson, I. and Marir, F. (1994a), "Case-based reasoning: A review", *The Knowledge Engineering Review*, Vol. 9, No. 4, pp. 327–354.

Watson, I. and Marir, F. (1994b), "Case-based reasoning: a categorized bibliography", *The Knowledge Engineering Review*, Vol. 9, No. 4, pp. 355–381.

Watson, I. and Oliveira, L. (1998) "Virtual Reality as an Environment for CBR", *Advances in Case-Based Reasoning: EWCBR-98 Proceedings*, B. Smyth and P. Cunningham (eds.), Springer-Verlag, pp. 448–459.

Wilke, W., Vollrath, I. and Bergmann, R. (1997) "Using knowledge containers to model a framework for learning adaptation knowledge", D. Aha and D. Wettschereck (eds.), *ECML'97 Workshop "Case-Based Learning: Beyond Classification of Feature Vectors"*, (http://wwwagr.informatik.uni-kl.de/~wilke/public/ecmlws97.ps.gz).

Wilson, D.R. and Martinez, T.R. (1997) "Improved Heterogeneous Distance Functions", *Journal of Artificial Intelligence Research (JAIR)*, Vol. 6, No. 1, pp. 1–34.

Yang, S.-A., Robertson, D., and Lee, J. (1993) "KICS: A Knowledge-Intensive Case-Based Reasoning System for Statutory Building Regulations and Case Histories", *4th International Conference on AI and Law, Amsterdam, Netherlands*, (http://www.dai.ed.ac.uk/groups/ssp/psfiles/soonae/icail-93.ps)

Chapter 5

AI for NDT data interpretation

Chapter 2 has presented the field of NDT, the typical techniques, the reasons for and possible ways of automating inspection in general and data interpretation in particular. Chapters 3&4 have presented various AI techniques capable of solving complex problems, including data interpretation. This chapter presents how AI techniques can be used for NDT data interpretation. First, however, other possibilities for use of AI in NDT are presented. This is followed by a general discussion of classification methods, however, from the possible feature extraction methods these suitable for NDT problems are presented. After that, the uses of various AI techniques for NDT data interpretation are presented, with examples of existing systems. Finally, the issues important for construction of successful AI systems for NDT are discussed, including impact of AI on NDT standards and regulations.

5.1 Possible uses of AI in NDT

In Chapter 2 reasons for automating NDT inspection have been discussed. This chapter presents how Artificial Intelligence techniques can be used for NDT inspection in general and NDT data interpretation in particular. For many simple problems, there is no need to use AI. Often, defects can be detected by simple thresholding of a properly preprocessed signal [Stepinski & Maszi, 1993]. AI techniques are more applicable for complex problems — generally, where a human operator is necessary for the data interpretation.

In NDT not only the interpretation of data, but the whole process of setting up the inspection, choosing a technique and its parameters requires expertise. Therefore, possible uses of AI in NDT are numerous, for instance:

- Choice of proper configuration for the inspection, like probes to be used, frequencies, etc. This can be done based on the goals and the physics of the inspection technique, but it also may include procedures required by applicable inspection norms. Use of AI for this purpose could contribute to the reliability and repeatability of inspection. It could also help the less experienced personnel to perform better quality inspections. Because new inspections are usually prepared based on experiences from previous inspections, CBR seems to be a suitable technique.
- Conditioning of data, for example, removal of complex noise and distortion patterns. People are quite good at spotting anomalies in complex background patterns and ignoring them, while automatic systems may have problems with this. The task of the automatic data interpretation, using AI or not, can be simplified by removing the irrelevant background pattern(s) consisting of noise, distortion, etc.
- Interpretation of data — mainly classification but also detection and characterisation. This is the subject of this chapter.
- Analysis of the results of inspections, especially, over a longer time span (trends, prediction).
- Modelling for prediction and for inversion. For example, ANNs can be used to model complex systems or electromagnetic and acoustic phenomena [Wang et al., 1997]. ANN models of phenomena while being a simplification of the physical equations describing them, are usually faster to compute and can be used for fast inversion and this way for defect characterisation. System models can be used for anomaly detection, for example, in acoustic turbine monitoring.
- Use of the knowledge acquired by learning AI systems for personnel training.

The remainder of this chapter will focus on the interpretation of NDT data, with emphasis on data classification.

5.2 Data interpretation

Generally, the term data interpretation applies to a situation where a meaning has to be assigned to some data. In NDT the data can have diverse forms, for instance, a collection of values, a 2-D image, a time-base signal, etc. The meaning given to the data can range from simple category classification, to complex characterisation comprising in addition to the defect type also, for example, the location, size, and other parameters of the defect.

Section 2.1 and 2.4.2 have already presented the three components of interpretation as present in NDT: detection, classification, and characterisation. In some applications the detection and the characterisation components may be missing, but the classification will always be present — at least of the defect/non-defect kind. Though the detection of regions of interest (the possible defects) in the data can itself be a complex problem, giving possibility to the use of AI techniques (e.g., [Chuang et al., 1998]), it will not be discussed here. Also little attention will be paid to the characterisation which, though it also provides possibilities for the use of AI techniques (e.g., use of ANN as models), is usually done using algorithmic procedures.

5.2.1 Distinguishing aspects of NDT data interpretation

An obvious question here is in what respect interpretation of NDT data differs from the general interpretation problems. Some of the distinguishing characteristics are:

- Complex data forms, consisting of signals, images, often requiring preprocessing to extract the relevant information.
- Uneven proportions of possible classes. Especially, the defects may constitute only a small fraction of all the indication types that have to be classified. A straightforward application of statistical classification techniques might lead to systems with low *overall* classification error, but which might also have low POD and low reliability.
- Often one has to deal with high variability of the inspection problem. It may be difficult to gather a good data set for training and testing.
- Sometimes, extra context or domain knowledge may be necessary to properly classify the indications, while in typical classification problems all the information is contained in the data and one can treat it just as numbers (for instance, for the classification problem illustrated in Figure 5.2 one needs no information about the problem represented by the data).
- Mistakes are costly, reliability is crucial. While in many applications misclassifications are not that serious as long as on average the performance is good, in NDT any misclassified defects may in principle have serious consequences.
- Cost is often a decisive issue in the eventual acceptance of the systems. The developed systems are judged based on their usefulness and not based on their technical sophistication.
- Systems will usually be used by different people than those who build them, this puts special requirements as to the ease of maintenance.

In many aspects, data interpretation problems encountered in medicine are similar to these in NDT. However, there the results of interpretation are almost always presented to a human for further evaluation and rarely actions are taken automatically based on the interpretation. Also, usually, significantly larger amounts of money are available for the development of automatic systems and for tests.

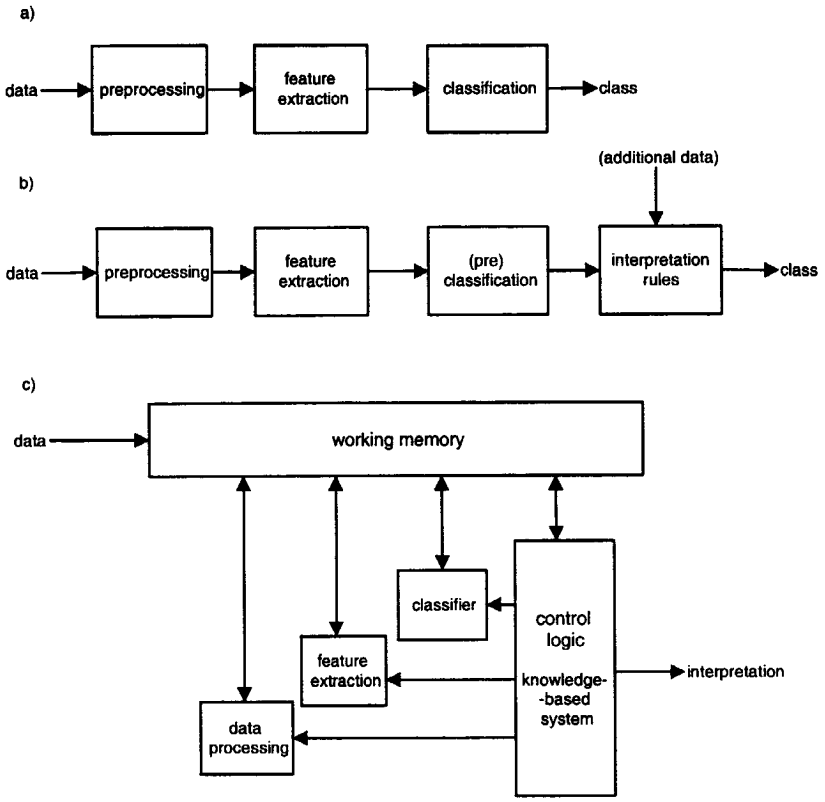


Figure 5.1: Some of the schemes for data interpretation: a) simple pattern recognition pipeline, b) pipeline with extra interpretation rules, for example, for verification and characterisation, c) nonsequential system (similar to blackboard architecture).

5.2.2 Methods of data interpretation

Usually, simple data interpretation is equivalent to pattern recognition. The actions taken during pattern recognition can be represented in the form of a "pipeline" consisting of three stages (Figure 5.1a): preprocessing, feature extraction, and classification.

The pipeline structure simplifies design of such systems. Each pipeline stage can be designed and implemented separately, making fine-tuning, changes, or extension of a system relatively easy. In more complex interpretation cases, the three-stage pipeline may be not sufficient, for example, when results of the classification stage are occasionally ambiguous or not complete. In such case a fourth stage, containing a set of rules, may do the final classification (Figure 5.1b). The fourth stage can also be used for precise data characterisation according to the determined data class.

In some situations the whole recognition and interpretation process may be under control of a knowledge-based system. The pipeline paradigm may then no longer be applicable. Signal processing, feature extraction, and classification can be done whenever the controlling system deems it appropriate (Figure 5.1c). Such a configuration is similar to blackboard systems [Engelmore & Morgan, 1988], especially if processing is explicitly data driven.

5.3 Pattern recognition

This section presents the three stages – preprocessing, feature extraction and classification – present in almost any pattern recognition system, from a simple pattern recognition pipeline to a complex knowledge-based classifier, as shown in Figure 5.1. The examples given in this section, for example for feature extraction, are mostly related to the test cases of the ANDES project.

5.3.1 Preprocessing

The preprocessing stage takes the raw data, as for example, registered by the sensors, and transforms it into a form better suited for further manipulation and interpretation. Also, irrelevant information like noise or distortion may be removed. It simplifies the task of the other stages of the pattern recognition.

The most common form of preprocessing is removal of noise. Sometimes, it can be accomplished by removing certain frequency components from the signal using, for example, lowpass, bandpass or highpass filters. However, such filters may cause some problems. Frequency filters have undesired side effects if the noise overlaps in the frequency domain with the signal from, for instance, the defect which is to be interpreted. Nonlinear phase filters may distort signal from defects making interpretation more difficult [Stepinski & Maszi, 1993]. If the noise is not random, one can try to model it and subtract from the signal. For example, Benoist et al. [1990, 1995] describe a system for EC signal interpretation in a presence of strong pilgrim noise. An adaptive filtering technique is used to remove the noise (which lies in the same frequency band as the defect signal). First, parameters of an autoregressive model of the noise are estimated and then the modelled noise is subtracted from the signal.

Usually, human operators are good at ignoring noise or distortion in the signal. If one can discover the heuristics that they use in a given problem, one can build a knowledge-based system which accomplishes the same. For example, Hellyar et al. [1995] describe a knowledge-based system for the removal of ocular artefacts from EEG signal.

A common preprocessing method encountered in EC testing involves mixing of signals of various frequency as described in Section 2.2.1. When a defect exists in a neighbourhood of a construction element, as for example a baffle plate, then the EC signal shape can be very complex making determination of the type and size of the defect difficult. One could try to make a system that could recognise defects directly from such signals, as done in [Lord & Satish, 1984; Yan & Upadhyaya, 1996], but this makes the interpretation system unnecessarily complex. The interpretation task can be simplified when the undesired indications are removed by signal mixing. Frequency mixing has a disadvantage that apart from the baffle signal, also the defect signal is suppressed. When a signal is processed digitally, this problem can be partially solved by doing the mixing only when it is necessary, i.e., when a baffle signal is detected. This implies that the sequential pipeline scheme is no longer followed, but the preprocessing is done only when needed.

Often, preprocessing has as a goal transforming the data into a form better suitable for the interpretation. For example, in the rail inspection system described in Chapter 7, the first part of preprocessing consists of transforming the echo indications contained in the data file into a 2-D rail image, in which the indications are correctly positioned over a rail-outline according to the physics of the acquisition system. This image can be then interpreted by a human operator. To facilitate interpretation by a computer, further preprocessing in a form of image segmentation is necessary.

5.3.2 Feature extraction

Usually, the original data has too many dimensions (parameters) to be directly classified (this is clearly a case for images). The data dimensionality has to be reduced, for instance, by means of feature extraction. The extracted features can range from simple (usually statistical) characteristics such as a maximum value, mean, variance, or other moments (for a signal), to more complex, like

position and orientation of lines, circles, etc. in an image. The extracted features are usually grouped into a feature vector which forms an input to a classifier.

Within NDT, one often comes across feature extraction for time-series signals (e.g., A-scans from ultrasonic testing). For such signals various features can be extracted. For example, the ICEPAK software package by Tektrend (www.tektrend-intl.com) can extract 108 features [Lacasse et al., 1988]. ICEPAK calculates features in five domains: time, frequency (power and phase), cepstral[1] and autocorrelation. However, usually only a few of these features are necessary to distinguish between various categories. The relevant features can be determined using a-priori knowledge about the data or by statistical analysis of the data.

Choice of features used by knowledge-based systems will usually be based on the features actually used by an expert during interpretation. This means that the features used will have some easily interpretable meaning, and not be just numbers resulting from applications of some transforms to (parts of) data. This also means that in principle it is possible to analyse these systems without actually testing them on data.

5.3.3 Feature extraction in images

Feature extraction is particularly complex in case of images consisting of pixels (vector images are generally easier in this respect). One can treat an image as a whole and calculate global features for the entire image. Another approach is to look for smaller constituent elements in the image. The simplest way to look for some structures in such data is by means of clustering; this technique is presented below.

When complex features have to be extracted, knowledge-based techniques can be used to look for features in data in an intelligent way. For example, den Hartog [1995] describes an application where an image of a utility map is searched for constituent elements (pipelines, buildings) using knowledge stored in a semantic network.

Clustering. Clustering is used when data has to be divided into meaningful groups. A group may, for example, consist of all the elements which are close (according to some distance measure) to one another. Clustering may be used in feature extraction, but also in data preprocessing (e.g., converting a pixel image to a vector image), and for classification, for instance, K-means clustering as shown in Section 5.3.4. Some generally applicable clustering techniques are:

- *Leader algorithm* [Hartigan, 1975] — Generally, this algorithm assigns every point to the nearest cluster. If the distance to that cluster is greater than some predefined value, a new cluster is started. Different distance measures are possible, but usually the distance to the median of the cluster is used. The algorithm is fast, but the results depend on the sequence in which the points are processed. This sort of algorithm is used in the cluster discovery network ART1, see Section 3.3.2.
- *K-means clustering* — The algorithm partitions a given data set into K subsets (clusters) in a way that minimizes some performance index F, for example, a sum of squares of distances between the cluster points and the cluster centre (described in [Udpa & Lord, 1984] with an example of use for classification). This is an iterative algorithm and the number of clusters K has to be specified beforehand. This type of algorithm is also represented by the Kohonen network for vector quantisation, see Section 3.3.2.
- *Minimum-spanning-tree (MST) clustering* — This clustering is done by constructing a minimum spanning tree for all the points (a minimum length graph connecting all the points, see, for example, Figure 7.12) and then breaking it up, for example, on the longest edges. When some

1. The complex cepstrum for a sequence x is calculated by finding the complex natural logarithm of the Fourier transform of x and then the inverse Fourier transform of the resulting sequence [Oppenheim & Schafer, 1975].

modifications are added to the algorithm, it is capable of separating quite complex clusters [Murtagh, 1985].

Some types of clustering algorithms that look for clusters of points having a certain shape, like line or arch, and thus applicable to images are:

- *Hough transform* [Gonzalez & Woods, 1993; Hopgood et al., 1993] — This algorithm searches for collinear clusters of points by means of transformation into polar coordinates. This method is often used to look for lines in pixel images, but can also be extended to look for arbitrary shapes [Ballard, 1981].
- *Variants of K-means clustering* — For example, Gustafson-Kessel [Krishnapuram & Freg, 1992], which constructs clusters that minimise the sum of the moments of inertia of all the clusters; thus, finding linear or hyperplanar clusters. C spherical shells [Krishnapuram et al., 1992] is similar to Gustafson-Kessel but searches for circle-like (or spherical) clusters in data.
- *Hopfield-net based algorithms* — Hopfield-type neural network with appropriately defined energy function can be used, for instance, to simultaneously fit several lines (or curves) to a set of points [Kamgar-Parsi & Kamgar-Parsi, 1990].

Many clustering algorithms can have the following variations [Krishnapuram & Keller, 1993]:

- *Hard* — This variation performs exhaustive and exclusive partitioning of data points. Each data point can belong only to a single cluster. The clustering quality deteriorates when spurious points (not belonging to any cluster) are present in the data.
- *Fuzzy* — Each data point can belong to more than one cluster. Cluster membership is represented by a number in the range [0...1]. A constraint is imposed requiring that the sum of the cluster memberships must be equal 1.
- *Possibilistic* — In this method the membership values can be interpreted as degrees of possibility of the points belonging to the clusters. It is not required that the sum of membership values be equal to one; therefore, any spurious data points will have all membership values very small.

In many algorithms, it is required that the number of clusters to search for is specified beforehand. However, such algorithms can always be transformed into iterative ones that discover the optimal number of clusters using some validity measures [Carkhuff & Udpa, 1987; Krishnapuram & Freg, 1992].

Feature extraction for shapes and curves. Image clustering algorithms give clusters of points as a result. If explicit line or other shape clustering was done then the clusters are already precisely described. Otherwise, the individual clusters have still to be characterised. Usually, the goal is to determine several parameters describing the shape of a group of points. The simplest parameter of a shape is its size, but already the shape orientation may be difficult to determine unambiguously. Sometimes, it is desired to derive features that are transformation (rotation, translation, and scaling) invariant.

A special kind of shapes are curves (a sequence of points), for example Lissajous curves from Section 6.1.3. Curves are usually described by coordinates of constituent points, but other representations, like first/second derivative, direction, (cumulative) curvature as function of length, etc., may be more suitable for feature extraction.

According to [Udpa, 1988] methods of shape-describing-feature extraction can be divided into parametric and nonparametric. The parametric methods are characterised by the fact that it is possible to reconstruct a curve from the derived descriptors. The nonparametric methods are one-way — it is not possible to precisely reconstruct the curve from the derived descriptors. Table 5.1 lists a number of methods that can be used to obtain features for shapes and curves [van't Hoff et al., 1996].

Table 5.1: Several methods to obtain features for shapes and curves [van't Hoff et al., 1996]. (One can apply methods suitable for curves to shapes after the edges of the shape have been determined.)

Method	Used for	Possible to reconstruct original	Remarks	Example of use
Fourier coefficients	curves	yes, (approximately if few coefficient used)	Usually less then 20 Fourier coefficients are enough to describe fairly complex *closed* curves. There is also direct relation between values of the coefficients and shape characteristics.	[Stepinski, 1990&1991], [Dekking & van Otterloo, 1986]
Fourier descriptors	curves	no	This method was originally used in character recognition. It has also been widely used for EC curve description. Fourier descriptors are obtained by a combination of Fourier coefficients such that the resulting values are rotation, translation, scale, and start point invariant.	[Zahn & Roskies, 1972], [Udpa & Lord, 1984]
Tchebycheff coefficients	curves	yes, (approximately if few coefficient used)	They give better results than Fourier coefficients for non-closed curves.	[Brousset & Baudrillard, 1994]
Freeman chain code	curves	approximately	This is an approximate representation by a sequence of directed strokes having a limited number of possible directions (usually 8) and fixed length.	[Kang & Kwon, 1995]
curvature features	curves	no, if single features are used, yes, (approximately) if a downsampled curvature vector is used	From the curvature of a curve one can determine a number of discrete features which, e.g., correspond to significant changes in the curvature, like corners, ends. One can also use a full or downsampled curvature vector to describe a curve.	[Asada & Brady, 1986], [Jarmulak, 1997b]
wavelets	curves	yes, (approximately if few coefficient used)	The low frequency coefficients of the wavelet transform of a curve can be used to describe it. Similar in applications to Fourier coefficients.	[Wuch & Laine, 1995]
B-splines	curves	yes, almost exactly	These are good for curve representation but a comparison and classification of curves is difficult because B-splines with different control points can have the same shape.	
Zernike moments	curves & shapes	no	invariant to rotation	[Khotanzad & Lu, 1991]
moment invariants	curves & shapes	no	Moments can be derived which are invariant to rotation, translation and scaling.	[Hu, 1962]
combination of nonparametric features	curves & shapes	no	Examples of features that can be used are: number of curvature changes, amplitude, phase, fatness, thinness, magnitude of distortion, sphericity, etc.	

5.3.4 Classification

The classifiers considered here are either statistical classifiers or AI classifiers like ANNs and decision trees. Their common characteristic is that they are built based on the data (training set), thus, in principle no a-priori knowledge is used in their construction maybe apart from selection of the features in a feature vector. If no training data is available then a-priori knowledge about the classification procedure is necessary to construct, for instance, a rule-based classifier.

These classifiers can be distinguished into supervised and unsupervised classifiers (compare Section 3.3.1 and Section 3.3.2). The supervised classifiers require availability of a training set with every element assigned to some class. After training, such classifiers can assign pieces of data to their corresponding classes. The unsupervised classifiers are not trained beforehand. They distribute (cluster) a batch of data into a (predetermined) number of groups, which later have to be named. The disadvantage of most unsupervised classifiers is that they are only able to process batches of data, and therefore may be unsuitable for on-line applications.

The most popular classifiers are:

- *(Empirical) Bayesian classifier* [Schalkoff, 1992] — This classifier assigns a piece of data to the category to which it has the greatest probability of belonging. Theoretically, the method requires knowledge of probability distributions. In practice, the probability distributions are estimated from the training data.
- *(K-)nearest–neighbour classifier* — Nearest-neighbour classifier assigns a feature vector to be classified to the same class as the nearest feature vector in the training set. *K*-nearest-neighbour classifier finds *k* vectors nearest to the one to be classified and assigns it to the majority class. The main difference in the various forms of the nearest-neighbour classifier is the way the distances are calculated. One can, for example, use a Euclidean distance, cross correlation, Mahanalobis distance, etc. Also, different features can be assigned different importance factors (weights).
- *Linear discriminant function* [Fu, 1982] — is similar in architecture to a neural network with no hidden layers and with a threshold transfer function (Adaline).
- *Minimum-mean-distance classifier* [Khotanzad & Lu, 1991] — It calculates distances between a feature vector and the means of each class and assigns the vector to the nearest class.
- *Parzen-window classifier* — In this type of classifier the probability density function is approximated by a sum of kernel functions (usually Gaussians) having centres at the locations of the samples from the training data set. The data is classified into a class with the highest value of the probability density function. This type of classifier is a good approximation of the Bayesian classifier.
- *Artificial neural networks* — These are in many respects similar to various statistical classifiers (see Section 3.6.2). The most popular neural networks are MLP and RBFN.
- *(Fuzzy) K-means clustering* — This is an example of the use of clustering for classification [Udpa & Lord, 1984] (see previous section on clustering). In [Carkhuff & Udpa, 1987] it is shown, that a fuzzy version of this method makes it possible to detect spurious feature vectors, and thus improves classification results.
- *Decision trees* — assign the data to one of the classes based on a sequence of (usually binary) decisions with respect to the values of the attributes (features) describing the data. The pattern of all possible decisions can be represented as a tree where the nodes correspond to the decisions to be made and the branches represent the possible alternatives. Decision trees can be built using a-priori knowledge, but usually they are induced based on the data from the training set. The choice of the attributes inspected at each node is made using measures from information theory. The main goal is that the attributes used and the decisions made at each branch of the tree have the maximal discriminating power. The best known decision-tree types are ID3 and C4.5 [Quinlan, 1993].

Classifier comparison. Classifier comparison is not an easy task. The main difficulty is the fact that classifier performance depends on the type of the classification problem and on the amount of data that has been used for training. This means that a classifier which performs well on some classification problem may perform much worse on another type of classification problem. Schaffer [1994] presents a theoretical discussion of the fact that positive performance in some learning situations is offset by a negative performance in other. He presents a *conservation law for generalisation performance*,[1] which states that: *total generalisation performance over all learning situations is null; positive performance in some situations must be exactly balanced by negative performance in others*[2].

Still, given well-defined conditions classifier comparison makes sense. The classifier characteristics which are typically compared are the classification accuracy and speed (time necessary for

1. Generalisation performance is defined as the expected prediction performance on cases not present in the training set.
2. This is a typical "no free lunch" theorem, similar ones exist, for example, for function minimisation.

training and time it takes to do the actual classification). Sometimes, also, the difficulty of implementation or simplicity of a classifier (e.g., a decision tree) can be included in the comparison.

An extensive comparison of various classifier types can be found in [Lim et al., 1997]. The study compared 33 classifiers using 32 data sets. The classifiers were from three categories: statistical (9), decision trees (22) and neural networks (learning vector quantisation and RBFN). As far as the classification accuracy is concerned the best performer was the statistical POLYCLASS classifier [Kooperberg et al., 1997], closely followed by the QUEST decision tree classifier. The ranking of the classifiers is given in Table 5.2. Worth noting is the fact that the average classification errors are quite similar. Strange is the long training time for the RBFN, as for this classifier type fast training algorithms are available.

Table 5.2: Classifier comparison results reported in [Lim et al., 1997] (a selection).

best average error	classifier	CPU times
0.195	POLYCLASS algorithm by [Kooperberg et al., 1997]	3.2h
0.202	QUEST - decision tree derived from FACT, using univariate or linear combination splits	3.7min
0.204	logistic discriminant analysis	4.0min
0.208	linear discriminant analysis	10sec
0.215	CART decision tree	52sec
0.217	flexible discriminant analysis	3.8h
0.219	IND decision tree (multiple Bayes version)	27.5min
0.220	C4.5 decision tree	**5.0s**
0.220	LMDT decision tree	5.7min
0.227	OC1 decision tree with univariate splits	46.0sec
0.234	FACT classification tree with linear combination splits	8.0sec
0.257	RBFN	11.3h
0.269	learning vector quantisation	1.1min
0.270	CAL5 decision tree for numerical valued attributes	1.3h
0.281	nearest neighbour with Mahanalobis distances	20.0sec
0.301	linear discriminant with estimated covariance matrix	15.0sec

A less rigorous search for the best classifier can be followed, for instance, in a series of articles from Iowa State University, where various classifiers were used to classify, generally, the same data set consisting of Fourier descriptors for EC signals:

- [Udpa & Lord, 1984] — use K-means clustering,
- [Udpa & Lord, 1986] — use nearest-neighbour classification,
- [Carkhuff & Udpa, 1987] — present fuzzy K-means as an improvement over K-means clustering,
- [Udpa & Udpa, 1990] — compare K-means clustering and MLP, MLP gives a better recognition ratio and is faster,
- [Nair et al., 1993] — compare MLP with RBFN, RBFN achieves both better results and is faster.

Other articles that compare various classification methods are, for example:

- [Doctor & Harrington, 1980] — compare linear discriminant, empirical Bayes, and nearest-neighbour, and concludes the nearest-neighbour classifier to be the best.
- [Chapman et al., 1991] — choose K-nearest-neighbour above linear discriminant, minimum distance, and empirical Bayes.

- [Khotanzad & Lu, 1991] — conclude that MLP performs better than minimum-mean-distance, and similar as nearest-neighbour for "clean" data; for noisy data, however, the MLP becomes clearly the better alternative. A significant advantage of the MLP is a significantly smaller number of computations necessary during the runtime, in comparison with the nearest-neighbour classifier.
- [Wielinga et al., 1994] — compare nearest-class and nearest-neighbour classifiers with an MLP type neural network classifier on a simple classification problem. The nearest-class and MLP classifiers perform similarly. Both are better than the nearest-neighbour classifier.

From the above mentioned articles one could conclude that the MLP and RBFN seem to be the best classifying tools, at least for the NDT problems which were the subject of test. The RBFN seems to be the better alternative because of a significantly shorter training time. However, the study done by Lim et al. suggests that decision trees (or the POLYCLASS classifier) might give even better results.

Still, in the light of the conservation law mentioned at the beginning of this section, one should remember that: *every demonstration that the generalisation performance of one algorithm is better than that of another on a test suite is an implied demonstration that it is worse on an alternative suite* [Schaffer, 1994]. Thus, any statements about classifier superiority hold only for well-defined problem subsets and controlled environment in which the classifier will be used.

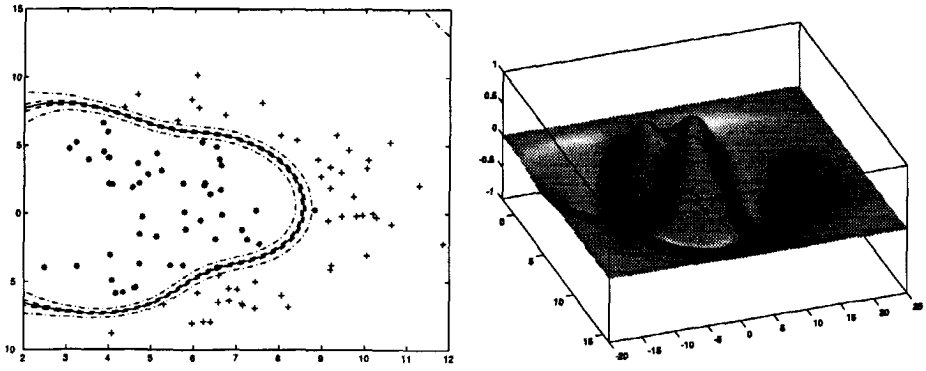
Problems with the use of classifiers. Obtaining a successful classifier depends on two main factors:

- The problem itself, or rather its representation as a feature vector, should be classifier tractable. In the feature space the points representing data from the same class should be closer to each other than to the points representing other classes. Wherever there is an overlap, classification errors may be made.
- The training set has to be representative for the data that will be encountered during the future use of the classifier. If this is not the case then any data not sufficiently similar to the data from the training set may be incorrectly classified.

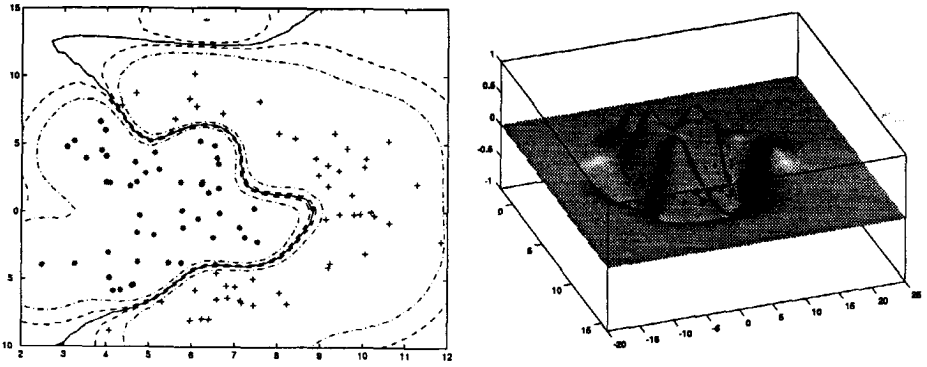
The main problem, as far as their application in NDT is concerned, related to the use of classifiers is how they cope with unknown data, for instance, signals significantly different from signals in the training set. Various types of classifiers cope differently with such data, for example:

- Fuzzy K-means clustering will assign to such cases low probability of belonging to some class, this way it can be easily recognised as unknown. However, clustering is not practical for on-line systems.
- Nearest-neighbour methods may be constructed to return not only the class of the nearest neighbour but also the distance to that neighbour. Some reject threshold may be used to determine if the data is to be classified as unknown.
- MLP (or RBFN) classifiers have in common that their behaviour for feature vectors that lie outside the training set area is not well defined. Thus if presented with unknown data such classifiers will return one of the trained classes as the result, even if the data is not similar to that class. This problem can partly be solved by using redundant output nodes, like in [Pratt & Sansalone, 1991].

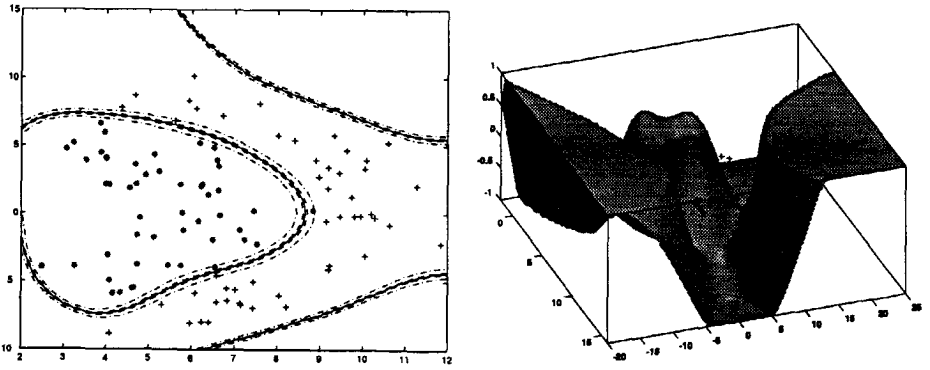
The problem of unpredictable response to unknown data and also the, simpler to detect, problem of overtraining, are illustrated in Figure 5.2. Data from two classes is to be classified: a class marked by * is half encircled by a "banana-shaped" class whose members are marked with +. An RBFN classifier as described in [Chen et al., 1991] is used for the classification. Three classifiers are constructed, differing in the width of the RBF kernels. Figure 5.2a shows the best classifier obtained in the experiment. Figure 5.2b shows the classifier which gives better results on the training set; however, from the complex border between the classes it is clear that the classifier fits the training data



A good classifier built using RBF kernels with covariance equal 1x covariance of the data: 1% error on the training set, avg. 4.2% error on the test sets.



A classifier overfitting the data. RBF kernel covariance equal 0.5x covariance of the data: 0% error on the training set, avg. 4.9% error on the test sets.



A classifier with unacceptable extrapolation. RBF kernel covariance equal 2x covariance of the data: 1% error on the training set, avg. 4.3% error on the test sets.

Figure 5.2: Three classifiers for the same data set. The left column shows the data set and the class boundaries (according to the classifiers). The right column shows the classifier function — the threshold is set at 0, class * is positive, class + is negative.

rather than the underlying class distributions. This can be detected without looking at the class boundary by using a test data set, for which the classification error is higher. Figure 5.2c shows a third classifier which performs quite well both for the training and the test data set; however, a visualisation of the classification function makes it clear that this classifier would make serious mistakes on outlying data from the + class. The extrapolation made by the classifier is unacceptable. The problem is that this condition is difficult to detect, certainly if the available data set is limited and the problem dimension increases (which makes the classifier function more difficult to visualise). Also, using reject thresholds to improve the reliability would not be feasible for this classifier. One can detect the danger of unreliable extrapolation if all the training data is stored and before the classification the distance to the nearest data exemplar is determined, as in nearest-neighbour and other instance-based techniques.

5.4 Use of various AI techniques in NDT data interpretation

5.4.1 Statistical and ANN classifiers

For simple NDT interpretation problems a vector of features can be directly used to classify the NDT data into the given classes according to the simple pattern-recognition pipeline from Figure 5.1a. Examples, which claim successful use of such classifiers are numerous, for example, [Berry et al., 1991; Lunin & Gomonov, 1994; McNab & Dunlop, 1991]. In more complex cases, the initial classification results may have to be processed further until they lead to the conclusion, or they may make the system decide to look for other features in the data for further classification, see Figure 5.1b&c.

Within the ANDES project several reported uses of the simple pattern-recognition pipeline for the classification of eddy-current data have been analysed. Most of the articles on this subject deal with a relatively trivial task of classification of the EC data obtained from simple numerical models [Lunin & Gomonov, 1994] or calibration defects [Udpa & Udpa, 1990]. Such systems fit neatly into the standard schema of a pattern-recognition pipeline, requiring hardly any use of logic to guide the whole interpretation process. Moreover, most of the articles are concerned primarily in the study of various signal parametrisation and classification methods. Few articles describe systems used in inspection practice.

Most of the reported systems for EC signal classification make use of Fourier descriptors as feature vectors. The popularity of Fourier descriptors as features for the description of EC Lissajous curves began with the articles by Lord & Satish [1984] and Udpa & Lord [1894]. Some systems use essentially the same approach to signal classification but choose different features to describe the signal, for example [Brousset & Baudrillard, 1994] make use of Tchebycheff coefficients in their SIAM system for checking rivet connections in aeroplane hulls for cracks. These and other systems are listed in Table 5.3.

A standard approach to signal classification is illustrated in [Chapman et al., 1991]. Chapman et al. describe a system for crack detection in an aircraft engine compressor disk. The system was built using the ICEPAK software. Because ICEPAK can only process one-dimensional signals, the analysis is not performed in the phase plane (thus one does not look at the shape of Lissajous curves), but the resistance and inductive reactance signals are processed directly. The useful features were automatically determined using ICEPAK which can extract more than 100 various features from the signal and then analyse their discrimination properties. The important features were found to be related to the signal amplitude and partial power in the power and autocorrelation domains. After the features are extracted from the signal, the feature vector is classified using several techniques (linear discriminant function, K-nearest-neighbour, empirical Bayesian, and minimum distance), of which the K-nearest-neighbour has been found to give the best results. The reported results are good, possibly because the specimen to be inspected is of high manufacturing quality, making the detection problem well defined (the defects are repeatable).

Table 5.3: A number of simple NDT data classification systems.

Reported in	Classification method	Application
[Lord & Satish, 1984], [Udpa & Lord, 1984]	K-means clustering	determination of the distance of a hole from a baffle plate from a single-frequency EC signal
[Udpa & Lord, 1986]	nearest neighbour	determination of amount of magnetite build-up under a baffle plate from a single-frequency EC signal
[Carkhuff & Udpa, 1987]	fuzzy K-means clustering	recognition of defects from 5 different classes (EC data)
[Udpa & Udpa, 1990]	feed-forward backpropagation-trained neural network and K-means (for comparison)	recognition of defects form 4 classes (through-wall hole, axisymmetric OD slot, flat-bottomed hole, and axisymmetric ID slot) based on EC data
[Lunin & Gomonov, 1994]	as above	recognition of defect from 5 classes (baffle, inside groove, outside groove, baffle & groove, baffle & magnetite) based on EC data
[Brousset & Baudrillard, 1994]	MLP	classification of EC signals (detection of cracks) from inspection of rivet connections in aeroplane hulls
[Nair et al., 93]	RBFN	sizing of defects based on ultrasonic signal (A-scan)
[Chapman et al., 1991]	K-nearest-neighbour	classification of EC signals from inspection of compressor disks

5.4.2 Expert systems

Classification can also be done using reasoning implemented in a set of rules. The classification rules can be readily available as a part of existing theory, or as, not so well defined, expertise. Rules can also be obtained from data using the same or similar techniques as for inducing decision trees.

For a simple system it may be enough to hard code any rules necessary to evaluate the data. For example, in [Pratt & Sansalone, 1991] a simple system for impact-echo-signal interpretation is described where the output of an ANN classifier is judged for a presence of any anomalies using hard-coded logic.

Benoist et al. [1990, 1995] describe a SOCRATE system for inspection of steam generator tubing. The whole system is built around a rule-based expert system. Apart from the direct use for signal interpretation, the rules can launch external procedures, for example, a noise removal procedure can be started when the signal-to-noise is below some threshold. The system can simultaneously process data from multiple frequencies (the example given in the article is for three frequencies). It tries to exploit complementarity of information contained in all the channels for accurate defect interpretation. The degree of coherence of the information in the channels is used to determine interpretation reliability. An example of rules used in the system is given in Figure 5.3.

The authors draw the attention to the relative ease with which the rule-base (which was written for a 900MW PWR SG) can be converted to other steam generators by changing the phase shift laws. Still, this requires rewriting the rule-base for (almost) every heat exchanger that has to be inspected.

For the purpose of the ANDES project a number of expert systems for EC data interpretation were analysed: the already mentioned SOCRATE, a similar EXTRACSION [Georgel & Zorgati, 1995], the Dodger system [Levy et al., 1993] (presented in more detail in Section 5.4.3), and a hybrid system described in [Kang & Kwon, 1995] (presented in Section 5.4.5). What is characteristic is that all four systems have been developed for nuclear power industry. This is not a coincidence. Development of successful expert-system applications requires, first of all, a well-defined problem domain – inspection of steam generators in nuclear power plants is such a domain. There is only one type of heat exchanger that has to be inspected and the inspection procedures are well standardised.

```
Rule R7

IF
   the phase shift of the signal at frequency F1 is < -140°
AND
   the phase shift at frequency F2 is < -120°
AND
   the phase shift of the signal at frequency F3 is < +120°
THEN
   the deformation is impact-induced.

Rule R8

IF
   it is a multiple defect
AND
   the depth indicated by channel F1 corresponds to a shallow internal loss of matter
AND
   the depth indicated by channel F2 corresponds to a near 100% defect
THEN
   there are two superposed defects of the shallow internal and external loss of matter.
```

Figure 5.3: Example of the rules used in the SOCRATE system.

Because there are standards the knowledge acquisition is simplified. Besides in the nuclear industry there are funds available to finance such projects.

An important disadvantage of expert systems for NDT data interpretation is that they require repeatable conditions to perform well. However, this is rarely the case, as most industrial NDT inspections are performed on a variety of installations with various procedures.

5.4.3 Reasoning with uncertainty

In the case of data interpretation in NDT, often no absolute statements over presence or absence of a defect can be made. Frequently, even a human expert may have doubts about giving a clear-cut diagnosis. Sometimes, extra measurements may be necessary to confirm initial suspicions. If a knowledge-based system was only able to give crisp answers, then it might result in its limited acceptance. Especially, when the conclusion turns out to be wrong, it is important to know whether the system was 100% or 55% certain that the conclusion was true. Apart from the uncertainty of the diagnosis, the knowledge of experts is full of approximate heuristics and ambiguities. Moreover, it is often expressed in linguistic, difficult to quantify terms. Because of this, it is advantageous if knowledge-based systems used in NDT data interpretation can model incomplete and uncertain knowledge and can reason with uncertainty. Rajagopalan et al. [1996] consider it even a necessity to incorporate reasoning with uncertainty in knowledge-based systems for NDT — they suggest use of a combination of fuzzy logic and neural networks.

An example of an NDT data-interpretation system which is capable of reasoning with uncertainty is DODGER [Levy et al., 1993] for eddy-current data interpretation. This is a significantly complex system which can operate both in an interactive mode (supporting the operator) and automatically. The automatic analysis process is guided by a set of rules. Rules manage the procedures that manipulate the data (e.g., signal mixing). The shape of EC signal in the impedance plane is processed, removing noise, spurious small loops, etc. It is then classified depending on the number of curvature changes (e.g., "8" or "V"). Extra parameters like the amplitude, phase, fatness, distortion, etc. are calculated. These parameters are then analysed using fuzzy membership functions.

For each diagnosis outcome a hypothesis is created. The believability in each hypothesis is updated as the data is analysed. The evidence is combined using Dempster-Shafer theory of evidence (see Section 3.5.1). This method provides an output in a form of belief intervals, which give good indication of the system's certainty in a given diagnosis. The knowledge about material type and manufacturer, heat-exchanger geometry, and other important circumstances is expressed in a set of constraint rules, which increase belief or disbelief in certain diagnostic outcomes.

The expert system can detect possibility of multiple-cause interactions and respond accordingly; however, in such cases the belief in the diagnosis is often low. DODGER is also able to include data from the previous measurements at the same site in the evaluation process. Also, data from neighbouring tubes can be used to reason about external deposits (as these frequently encompass several adjacent tubes).

The confidence in the system is increased by the fact that it can give a comprehensive trace of its reasoning to justify a conclusion. The system also detects situations where it can give no reliable diagnosis. Though, such widespread use of reasoning with uncertainty has clear advantages, such systems are difficult to build and difficult to validate. Complex, knowledge-based systems can be quite successful even if they do not implement reasoning with uncertainty, for instance ARBS described in [Hopgood et al., 1993] (see Section 5.4.5). Still, it is useful when a system can return a measure of confidence in the reported classification, but this can often be achieved using simple means, for instance, many classifiers can be designed to return a measure for class membership.

5.4.4 CBR

Case-based reasoning is a relatively new AI technique and not so well-known as ANNs or expert systems. The unfamiliarity may be the reason why until now CBR has hardly been used in NDT; because, as far as its capabilities are concerned, CBR seems to be well suited for NDT data interpretation. CBR is suitable for weak-theory domains and many practical NDT data interpretation problems are to a large degree weak-theory-domain problems. Also, as Section 4.4 has pointed out, CBR is reliable which is an important factor in NDT. The construction effort for CBR systems can be relatively low and their maintenance is easy, certainly when compared with rule-based systems.

Moreover, NDT data interpretation itself is to a large degree CBR-like. Though in some interpretation problems an inspector uses a domain model to interpret and characterise data, for example, in simple ultrasonic pulse-echo inspection; in many NDT problems, the underlying physical phenomena are too complex to be handled entirely by an inspector. What is done, is that an inspector interprets new signals based on the signals he has previously seen during training and during other inspections. A calibration phase is used to provide examples of what the signals from the particular inspected object will look like and to provide guidance for their characterisation. General domain knowledge is used only to explain problematic cases, and even then one may decide to switch to a different technique from which the data is simpler to interpret, for example, changing from eddy-current to ultrasonic inspection.

It seems that (apart from the research described in this thesis) the only research explicitly dealing with the use of CBR for NDT data interpretation has been done by Perner [1993, 1996, 1998] and Oatley, Tait and MacIntyre [1998]. Perner has investigated a possibility of using CBR for image interpretation in general, and interpretation of ultrasonic B-scans of welds in particular. However, this research did not lead to a system to be used in NDT practice[1], though CBR has been shown as suitable for the task of image interpretation. Oatley et al., have developed a CBR system for vibration analysis. The goal of their system is to support the operator in analysis of vibration data by retrieval of relevant records from a database of previous vibration measurements

Apart from that, a large number of CBR systems for classification have been built (e.g., PROTOS [Porter et al., 1990]) some of them for problems similar to these encountered in NDT, for instance, the system for classification of sonar images described in [Aha & Harrison, 1994]. These can provide guidance in developing CBR systems for NDT data interpretation. NDT problems have some characteristics which provide new challenges to CBR. One of them is handling data in form of signals, curves, images, etc. Another is the need for reliable automatic evaluation of the classifications made by the system, because the data automatically classified will generally not be

1. A choice for the use of models and simulations was made instead.

subject to any further verification. Other challenges are: the retrieval speed and handling of noise in the data.

Apart from learning by acquiring new cases, some CBR systems also learn from failures, which in case of CBR for NDT would be false suggestions made to the operator. However, the operator will typically have no time to provide more information concerning these failures other than entering the correct classification/interpretation, so the possibilities of learning from them will be limited.

Chapters 6 & 7 present two CBR systems for NDT data interpretation that have been developed within the ANDES project.

5.4.5 Hybrid systems

McNab et al. [1994] suggest the use of a combination of techniques for NDT data interpretation. Concentrating on a single technique (like a rule-based system) may result in a complex but inflexible system that may be difficult to maintain. A better approach is to identify various subproblems in the data interpretation and then solve each of them using the best technique. One can use, for example, a neural network to identify general indication types, algorithmic techniques can be used in defect characterisation, and a rule-based system can be used to generate hypotheses, to choose the best applicable technique available and to verify the results. Rajagopalan et al. [1996] suggest that a combination of fuzzy logic and neural networks can be used to build a system capable both of handling uncertainty and of learning. They consider a blackboard architecture as best suited to combine a number of techniques in one application.

A hybrid system for EC signal interpretation is described in [Kang & Kwon, 1995]. The "front-end" processing is done by a combination of syntactic and neural pattern recognition. A KBS is responsible for "back-end" processing. The front-end, which detects certain classes of signal shapes and gives initial indication about defect harmfulness, is built by integrating fuzzified syntactic pattern recognition and neural network concepts. The Lissajous curves are initially described by a fuzzified Freeman chain code. The syntactic recognition is used for extraction of flaw-suspected event patterns. A neural network is used to facilitate the decision on whether an event is a harmful flaw. The results of the first stage are later processed by an object-oriented rule-based expert system. The article states that the productivity of the human inspector using the system has increased by approx. 120% and the confidence in inspection has improved by 47%.

In [Hopgood et al., 1993] a system called Algorithmic and Rule-Based Blackboard System (ARBS) for B-scan interpretation is described. The starting point for the interpretation by the ARBS is a B-scan of a weld in a steel plate, acquired with ultrasonic transducers placed at several angles. A series of lines is fitted through the echo points of an image by means of Hough transform. Some groups of lines are recognised as echoes from the surfaces of the plate and are excluded from further processing. Other groups of lines are potential indications of weld defects. The system analyses each suspicious group of lines using rules and neural networks. A suspected defect indication is further tested for occurrence of additional verifying information (shadow, reverberation echoes).

The development of ARBS went through three stages. In the first stage, there was a single rule base that was found to be too inflexible. The second generation made use of a blackboard-based system, which allowed for flexibility in combining various knowledge sources (procedural, rule-based, and, in version 3, neural networks). Hopgood et al. wrote the software from scratch, because, as they say, the available expert-system shells were considered not versatile enough. Also, the high cost of powerful, commercially available expert-system shells played a role in the decision.

The system developed for the URS described in Chapter 7 is also a hybrid system, combining a rule-based and a case-based classifier. A short discussion of integration of rule-based and case-based systems can be found in Section 7.3.2.

5.5 Developing AI systems for NDT

The previous sections of this chapter have presented how various AI techniques can be used for NDT data interpretation. The goal of this section is to present some considerations which have to be taken into account when choosing the best suited technique or techniques for a given NDT problem. The main factor defining the interpretation problem is the inspection technique used. However, this only indirectly influences the choice of AI technique for data interpretation. Factors more directly influencing choice of technique are the data characteristics and the knowledge about the interpretation. The two following sections present these factors. Then, their influence on the choice of AI technique(s) is discussed.

5.5.1 Characteristics of NDT data

The four characteristics of NDT inspection which are of the most influence on the (options for) choice of any technique for automated interpretation of the data are: homogeneity of the data, possibility of unexpected events in the data, availability of a representative set of data examples, and repeatability of the data:

- *Homogeneity of data* — Homogeneous data will be uniform in structure and composition, usually possible to describe with a fixed number of parameters. Homogeneous data is encountered in simple NDT inspection, for example, quality control during production. Inhomogeneous data will contain various combinations of indications from construction elements, defects and noise sources. An example of inhomogenous data are B-scan images as described in [Hopgood et al., 1993] or as encountered in the ultrasonic rail-inspection system described in Chapter 7.
- *Possibility of unexpected events in data* — Unless the inspection occurs in a well-controlled environment, there is always a possibility of occurrence of unexpected events in data. These can be caused, for example, by previously unseen type of damage. The amount of unexpected data is reduced if inspection is done periodically. An experienced operator can usually correctly interpret unexpected data, but some automatic classification systems may fail and misclassify the data.
- *Availability of a representative set of data examples* — Such a set may significantly simplify construction of any automatic interpretation system. The example data is normally obtained from calibration pieces, however, they often represent only the most common defects and are usually expensive to manufacture. Recently, more and more data is stored as digital inspection records, unfortunately the stored data is rarely fully classified, as this would increase the cost of inspection (generally, only the serious defects are given full description in the inspection reports).
- *Repeatability* — This refers to two aspects of inspection: similarity between objects that are inspected and possibility of maintaining constant inspection conditions (settings) for all the inspections performed. Interpretation of data in repeatable conditions will be simpler. Usually, inspection during or after manufacturing process will be repeatable. Another example of repeatable inspection is inspection of heat exchangers in power nuclear plants and inspection of aircrafts as these are well standardised. However, a large part of the NDT inspection done is not repeatable.

For homogeneous NDT data and repeatable inspection conditions successful automated interpretation systems can be relatively easily developed. Usually, they use standard techniques from statistical classification or AI. Design of automated interpretation systems for heterogeneous data coming from non-repeatable, small-volume inspections with little a-priori information about the pieces or constructions to be inspected is far more difficult.

5.5.2 Types of knowledge about data interpretation

Knowledge about NDT inspection that can be used for data interpretation can have a form of (official) guidelines and rules, physical models, expertise (experience) of NDT inspectors, and records from previous inspections:

- *Guidelines and rules* — Some inspection regulations clearly state the ways of signal interpretation and procedures that have to be followed. However, often, they assume good quality data. Noisy or otherwise distorted data often creates unclear/ambiguous situations where decisions have to be made.
- *Models* — Models of the inspection technique can sometimes be used for data inversion and thus precise defect characterisation. However, models do not always perform well if the data is distorted by noise or in some other way. For some inspection techniques only non-invertible models are available. Some models can be expensive to compute. Apart from modelling the system response to defects it is possible to model noise and to use these models for efficient noise removal where standard noise removal techniques do not work well. Models can also be used to generate data examples.
- *Expertise* — An NDT inspector bases his decisions in interpreting data to a large extent on his experience. Of course, knowledge as contained in subject handbooks is important, but it is the direct experience that makes him an expert. This means that large part of the knowledge how to interpret the data is unwritten.
- *Inspection records* — Detailed inspection records containing the inspection parameters, the acquired data, and the classifications can be an invaluable source of knowledge for data interpretation. Lack of inspection records can to a certain degree be compensated by using data from model simulations.

5.5.3 Choice of AI technique

Now that the data characteristics and knowledge types have been presented, their influence on the choice and use of AI techniques (ANN, ES or CBR) can be discussed. The influence of the data characteristics is summarised in Table 5.4. Table 5.5 summarises how the available knowledge is related to the three AI techniques. The specifics of the use of each technique are presented in the following.

Neural network classifiers. ANN or statistical classifiers impose strong requirements on the data and the inspection, however, when these are fulfilled then good, fully-automatic classification systems can be developed with relatively little effort. This is, for example, the case if the inspection is a part of a manufacturing process, where the inspected pieces and the possible defect mechanisms are well known and the whole NDT inspection is done in repeatable conditions. In such cases it is possible to collect (or manufacture) a set of defect pieces, which can be used to obtain a training set. There are some commercially available tools (like ICEPAK [Chan et al., 1988]) which can construct classifiers without any a-priori information, based only on the training sets of data. One has, however, always to remember about the limitations of this technique (as presented in Section 5.3.4); otherwise, serious misclassifications may go unnoticed.

Expert systems. In situations where statistical classifiers cannot be used, because of the complexity or inhomogeneity of the data, rule-based expert systems can sometimes be a solution. Complex images can be more readily described by rules than represented as simple feature vectors. Rules can be devised which cope with inhomogeneous data by, for example, triggering some specialised data-processing algorithms.

Construction of expert systems is facilitated if it is possible (at least approximately) to describe (model) expected signals from defect and non-defect pieces. If no models for the problem are available then the knowledge about the problem has to be acquired from an expert (an NDT inspector).

Table 5.4: Influence of various data characteristics on the use of AI techniques for NDT data interpretation.

AI technique / data characteristics	neural network classifiers	expert systems (ES)	CBR
homogeneity of data	representing the data as a feature vector in principle requires homogeneous data – inhomogeneous data may make it difficult or impossible to design an ANN classifier for it	better capable of coping with inhomogeneous data; however, the more inhomogeneous the data the more effort is necessary to construct a rule-base	the more homogeneous the data the faster the system will learn it, inhomogeneous data will require more operator interaction
unexpected events	unexpected events reduce reliability of the system – will usually be misclassified	ES will usually recognise them but the recognition ratio will decrease	will be recognised by the system as such and learned
training set available	crucial to the development of a classifier	may be useful for inducing rules and testing, not strictly necessary if an expert or an approximate model for the inspection is available	not strictly necessary, useful for testing and for constructing initial case-base
constant conditions	conditions of use may not differ from the conditions under which training set was obtained (unless the difference is compensated by preprocessing)	necessary, otherwise, the recognition ratio will fall	a CBR system should be able to adapt to changing conditions

Table 5.5: Relation between interpretation knowledge sources and AI techniques. Models can be used to generate data examples, these can be used in a similar way as inspection records.

AI technique / knowledge type	ANN (classifiers)	expert systems (ES)	CBR
guidelines and rules		a situation where an expert system would be most successful	
models	ANN can be used to learn models from the data if no physical model is available; even if physical models are available, ANN can be used because of fast computation	ES can be used to guide proper use of the models in defect evaluation or noise removal (e.g., determine applicability), ES can use models to test hypotheses	can be used to fine-tune case matching, but mainly to evaluate the correctness of the solutions (classifications) and in solution adaptation (if present)
expertise	used to choose feature set and to verify ANN results (e.g., determine the most "difficult" data cases)	"traditionally" typical situation where an expert system could be built; however, often, knowledge elicitation can be difficult	can be used to determine proper case representation and indexing, to design matching algorithm, and to design evaluation and adaptation stages
inspection records	classified data can be used to train ANN if only it can be represented in suitable form (feature vector)	can be used to automatically induce rules for an ES; can also be used for testing	inspection records can often be successfully transformed into cases and used in CBR

However, the knowledge possessed by the expert is often incomplete and not well formalised, which makes knowledge acquisition a difficult task for the knowledge engineer. The whole knowledge acquisition process is also a burden for the inspection company, especially a smaller one, as it prevents the expert from taking part in routine inspection tasks.

Rule-based systems have the advantage that they usually report when they are incapable of solving a problem (interpreting data). However, any irregularities in the inspection not foreseen during rule-base construction will significantly lower the recognition ratio.

Apart from the cost of knowledge acquisition, another disadvantage of rule-based systems is the difficulty of rule-base maintenance. Maintenance may be required when changes are made to the inspection system, the inspection procedures, or if differing constructions are inspected. Maintenance usually cannot be done by end-users.

Case-based reasoning. CBR methodology is well-suited for NDT data interpretation because CBR can cope well with variations in data which may be due to different objects being inspected, different inspection conditions, and different equipment settings. One reason for this is the ability of CBR systems to learn from the data classified by the operator and to adapt to new inspection situations. Another reason is the reliability of CBR systems (see Section 4.4). CBR can be used in situations where no reliable statistical classifier can be designed, and rule-based classifiers would be either inefficient or impractically complex.

In principle, CBR systems require that an operator be present during data interpretation. The operator has to provide the correct classification for the data that could not be interpreted by the system. This means that a completely automatic CBR system is, for most inspection problems, not feasible, but a typical NDT inspection almost always requires presence of an operator.

When implementing CBR systems, one has to able to define and implement methods for distinguishing between data from different classes. This is a more difficult problem than when constructing a simple data classifier, as the important parameters cannot be simply determined based on a set of examples. One has to have some a-priori knowledge about the important features that distinguish various data classes, as well as anticipate possible data forms that can be encountered during future inspections. This may make it necessary to use more features to describe the problem than a comparable classifier would use. When determining the data representation and the matching algorithms, tradeoffs may have to be made between desired reliability and the recognition ratio.

CBR will not function properly if noise dominates the data. Also, at least at present, speed is a critical factor. Speed of the standard computer hardware is just about sufficient to develop useful systems. Because CBR systems can learn on-line, precautions have to be made that no wrong cases are entered by an inspector; otherwise, CBR systems may fail.

Table 5.6 summarizes various aspects of the use of the three AI techniques.

Table 5.6: Various aspects of the use of the three AI techniques.

AI technique / aspect	neural network classifiers	expert systems	CBR
construction effort / cost	low if training set available, otherwise, collecting the training data may be expensive	usually rapidly increasing with the increasing desired recognition ratio, especially expensive if extensive knowledge acquisition is necessary	lower or comparable to that for an expert system; however, less knowledge acquisition is necessary
maintenance	the classifier has to be retrained	difficult for the end-user, practically all maintenance has to be done by the rule-base designer	usually confined to case-base maintenance, can be done by the end-user
reliability	if unexpected data occurs then low for most classifier types	high if no mistakes were made during knowledge acquisition	high
recognition ratio	high if the data is similar to the training set	depends on the number/complexity of the rules	depends on the homogeneity of the data, *increasing* during the use to high
speed	high for most classifier types	depends on the number/complexity of the rules, usually sufficient	slow because of case-base searching

5.5.4 Factors deciding about success

Chapter 2 already has discussed the subject of automating NDT data interpretation. From the requirement that the benefit of using such a system should be higher than the cost of introduction and use, a number of factors deciding about the success of NDT data interpretation systems can be derived. These are discussed in the following. Much of this discussion is applicable to non-AI data interpretation but the use of AI introduces special factors which have to be considered like:

(usually) higher complexity, non-algorithmic working of some techniques, and the possible difficulty in explaining achieved results, learning (an the problem of assuring correctness of the learned knowledge), and, sometimes, irrational approach to the use of AI techniques (mistrust or hype).

Increase in the speed of inspection. This is usually the first benefit one thinks about when considering automation of the inspection. However, this is not the only benefit and probably also not the most important. A correctly working system should always contribute to the increase of the speed of inspection by doing the interpretation faster than the operator does it. Still, at the moment, some of the AI techniques, like CBR, require much computation. A complex system, like the one described in Chapter 7, may be barely capable of achieving the desired speed given the currently (1998) available hardware. The problems with the speed are, however, mainly due to the amount of data and not due to the complexity of the interpretation task; therefore, they will cease to be problems as the computer hardware gets faster.

Increase in the reliability of inspection. Chapter 2 has defined reliability as depending on the POD and PFI. These can be positively influenced by a use of more sophisticated interpretation methods (not available to the inspector) like: simultaneous use of more data channels, use of models, large case-base with classified data. Generally, use of automation should lead to the increase in consistency and repeatability of interpretation which positively influences reliability. Reliability of an automatic system can, to a certain extent, be verified by conducting tests similar to the ones already done for NDT inspection techniques. However, it is in principle impossible to test how the system will perform if presented with unexpected data. Humans are good in handling, or at least in recognising, new unexpected situations. Some AI techniques, like ANNs, for example, may fail in such situations. CBR on the other hand is designed to recognise new situations, which makes it, in principle, a more reliable technique.

Better, more complete documentation of inspection results. Any automated data-interpretation system can in principle contribute to better documentation of results, especially when combined with an automatic scanning system which provides precise positioning data. When a system has certain "speed reserves," they can be used to characterise more defects than it is normally done (characterise also the small, non-critical defects). This information can be valuable, for example, to analyse the trends in the growth of the defect population.

Cost of the development. Like with any software system, the development cost includes the cost of design and implementation. However, development of AI system also involves the knowledge acquisition step, which may constitute a significant fraction of the whole development cost. Knowledge acquisition will usually consist of a combination of formalisation of the knowledge (expertise) about the inspection technique and the data interpretation, and of collecting data examples. The problem is that it is easy to obtain the knowledge about interpretation of a few typical defect types. An expert can easily recall them and the data can be easily obtained from the calibration. If a system is built using only this knowledge it may be unreliable and give incorrect interpretation which has to be double checked by the inspector, which, in turn, may lead to abandoning the use of such a system. Still if the system "knows" how to apply this limited knowledge to the *appropriate* data cases it can be useful even if its recognition ratio is low. Such systems can scan the data, classify the simple indications and leave the interpretation of the more complex data for the inspector.

Because, acquiring the full knowledge about data interpretation is often so difficult and/or expensive, learning AI techniques are attractive for NDT problems, especially if the learning can be easy controlled by the users, like in the case of CBR.

Cost of introduction and cost of use. The lowest cost of introduction and cost of use can be expected from a fully-automatic system that requires no operator interaction. Such a system would be attached to the acquisition hardware, would process the data, and would provide reports about detected defects. Such systems do exist, but only for very well-defined problems. Before they are put

into operation test runs are necessary to fine-tune them, the results have to be monitored from time to time, and any changes to the inspection may require complex adaptation of the system. However, much of NDT inspection is done in varying environments, which would mean that fine-tuning and adaptation are necessary for every inspection. This would make such systems unattractive for many applications.

Because fully-automatic data-interpretation systems are in practice not very flexible, an alternative is to use AI to assist NDT inspectors in data interpretation. An operator can take care of preparing the system for a new inspection, monitor the results, adapt the system when necessary, and the automatic system can automatically classify the data it can recognise and provide some classification suggestions for the user concerning the remaining data. The cost of introduction and the cost of use of such a system will be lowest if it fits well in the inspection procedure the operator is already acquainted with and the ways of data presentation, interpretation and reporting are similar to the ones already used. A way to achieve this is, for example, to use the calibration phase before an inspection for adapting the system to the new conditions. Because the calibration data seldom contains the full range of possible indications, it is advantageous if the system can learn from the new data encountered during the actual inspection. When some problems with the automatic classification are observed, it is important to be able to provide an easy-to-understand explanation for the reasoning behind the automatic classification. The explanations will be easier to understand if the working of the system itself is easy to comprehend. Again, CBR seems to fulfil these requirements quite well.

Cost of system maintenance. A specific part of the cost of the use of an automatic data-interpretation system is the cost of the system maintenance. It may include adapting the system to changing inspection conditions, for example, by retraining the classifier, or by changing the rules. For a learning system it may include checking the correctness and consistence of the acquired knowledge. The system may also be periodically tested (for example, by checking a selection of the classifications made by it) to determine if it works as expected. If serious problems are detected, one must be able to determine the reasons for them and then make the necessary corrections.

The user of a system should be able to do most of the system maintenance. Only in exceptional cases should it be necessary to have the system adapted by the developer.

The personnel. Often, when considering automation of NDT data interpretation, one thinks about the reduction of the number of personnel, or reducing the qualifications necessary for the personnel, as an important source of reduction of the cost. This may be the case if it is possible to develop a fully automatic system for a given inspection problem. However, as already noted, for most inspection problems only operator-assisting systems are feasible. In situations when normally several people were needed to do the data interpretation, introduction of a fast automated system may lead to a reduction of the number of personnel needed to do the interpretation.

However, as far as the necessary qualifications are concerned, it seems that when an automated system is introduced the operator has to understand at least the basic ideas behind the working of the system to be able to use it properly. Even better knowledge of the system and the techniques used is required from the person responsible for the inspection as a whole. The supervisor has to make the decisions about preparing system for new inspections, be able to handle properly in the case of problems, and, in general, be responsible for the maintenance of the system.

Commercialisation. For a system to become commercially successful it has to perform well, not only in the laboratory experiments, but also during field inspections. Still, even then the success is not guaranteed. First of all there has to be need for this type of system – the "pull" from the market. Sometimes, one can help to create it by having good system demonstrators, making the capabilities and possibilities of the new technologies known to the practitioners in the field. Also, any new sys-

tem should fit well into current inspection practices, so that a switch to the use of new techniques should be easy to make.

It would be unreasonable to expect that a newly developed automated system would right from the beginning perform perfectly when used to do real inspections. Various minor or even major adaptations to the software would certainly be necessary. For example, it might turn out that in spite of all the tests there are some kinds of signals where various defect types could be confused. Successfully solving such problems would require cooperation of the following parties:

- The user of the system, i.e., an inspection company. They would have to report any abnormal behaviour, as well as provide suggestions for system improvement. It is advantageous that the user has some idea of the way the systems works, fortunately, in case of CBR working of the system seems to be quite easy to comprehend. The user will normally have more understanding of the inspection problem than the designer of the automated data-interpretation system.
- The developer of the data-interpretation (sub)system who is responsible for translating problems reported by the users into the adaptations and extensions of the algorithms used in the system software.
- The inspection system manufacturer having knowledge of the inspection hardware and software.

Such a cooperation is difficult to realise. However, as in any commercial enterprise perspectives of financial gain from the use of a better system would certainly justify the efforts for its development. These gains should, first of all, be clear for the direct user of the system – the inspection company.

5.5.5 AI and NDT standards

Most of NDT inspection is regulated by official rules and codes prepared by regulatory bodies like ASME, Nordtest, etc. The purpose of these regulations is to guarantee safe operation of structures inspected according to the specified rules. Inspection companies comply with these regulations because:

- otherwise, they would not be considered for doing a particular inspection job,
- by complying with regulations they can be relieved from part of the responsibility for any damage caused by not detected defects,
- it makes the decision-making related to the inspection easier.

According to the existing regulations, before NDT inspectors are authorised to choose and perform tests and interpret results they have to be trained. After training they are given certificates of an appropriate level (I, II, III) which determines their level of competence. Generally, level III is authorised to choose the appropriate NDT test method for a given inspection problem. Level II is authorised to interpret the results, and level I is authorised to do the inspection (do the scanning, etc.).

The automatic data interpretation systems fall into the sphere of competence of the level II certified personnel [Papadakis & Mack, 1997]. This would imply that they would also have to be properly certified. Part of the test that the personnel has to pass involves interpretation of defects in test pieces. This type of test can easily be performed with an automatic interpretation system. However, the personnel also have to pass other tests which evaluate their knowledge of the inspection problem and the techniques used. Such a test cannot be carried out on data interpretation systems, therefore, correctness of the knowledge contained in a system should be also achieved by its proper design.

If an automated interpretation system were used for the inspection then the level II personnel would be directly using it. For best results the personnel should have at least some knowledge about the reasoning method behind the automatic interpretation. The level III personnel would have to be responsible for the proper configuration and maintenance. For example, they would have to

choose the proper samples to retrain a classifier or check the consistence of a case-base. This means that level III personnel training should include at least an introduction to the AI techniques used for data interpretation.

5.6 References

Aha, D.W. and Harrison, P. (1994) *Case-Based Sonogram Classification*, NRL Formal Report, NRL/FR/5510--94-9707, January 31, (NCARAI Report: AIC-93-041).

Asada, H. and Brady, M. (1986) "The curvature primal sketch", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 8, No. 1, January, pp. 2–14.

Ballard, D.H. (1981) "Generalizing the Hough Transform to Detect Arbitrary Shapes", *Pattern Recognition*, Vol. 13, No. 2, pp. 111–122.

Benoist, B., Cherpentier, C., David, B., and Pigeon, M. (1990) "Expert system designed to increase the reliability of steam generator tubings", *Proc. 10th Int. Conf. NDE Glasgow, UK, June 1990*, pp. 187–189.

Benoist, B., Gaillard, P., Pigeon, M., and Morizetmahoudeaux, P. (1995) "Expert system for the characterization of defect signals in steam generator tubes", *Engineering Applications of Artificial Intelligence*, Vol. 8, No. 3, June, pp. 309–318.

Berry, D., Udpa, L., and Udpa, S.S. (1991) "Classification of Ultrasonic Signals via Neural Networks", *Review of Progress in Quantitative Nondestructive Evaluation*, Vol. 10A, D.O. Thompson and D.E. Chimenti (eds.), Plenum Publishing, New York, pp. 659–666.

Brousset, C. and Baudrillard, G. (1994) "Neural Networks for Automatic Diagnosis: Crack Detection on Aircrafts", *6th European Conference on Non Destructive Testing, Nice 94*, pp. 507–511.

Carkhuff, M. and Udpa, S.S. (1987) "An improved defect classification algorithm based on fuzzy set theory", *Review of Progress in Quantitative Nondestructive Evaluation*, Vol. 6A, D.O. Thompson and D.E. Chimenti (eds.), Plenum Publishing, New York, pp. 791–798.

Chapman, C.E., Fahr, A., Pelletier, A., and Hay, D.R. (1991) "Artificial intelligence in the eddy current inspection of aircraft engine components", *Materials Evaluation*, September, pp. 1090–1094.

Chan, R.W.Y., Hay, D.R., Matthews, J.R., and MacDonald, H.A. (1988) "Automated Ultrasonic System for Submarine Pressure Hull Inspection", *Signal Processing and Pattern Recognition in Nondestructive Evaluation of Materials*, C.H. Chen (ed.), Springer-Verlag, pp. 175–187.

Chen, S., Cowan, C.F.N., and Grant, P.M. (1991) "Orthogonal Least Squares Learning Algorithm for Radial Basis Function Networks", *IEEE Transactions on Neural Networks*, Vol. 2, No. 2., March, pp. 302–309.

Chuang, S.-F., Basart, J.P., and Moulder, J.C. (1998) "The application of wavelets and fuzzy logic to eddy current flaw detection in steam generator tubes", *Review of Progress in Quantitative Nondestructive Evaluation*, Vol. 17A, D.O. Thompson and D.E. Chimenti (eds.), Plenum Publishing, New York, pp. 775–781.

Dekking, F.M. and van Otterloo, P.J. (1986) "Fourier Coding and Reconstruction of Complicated Contours", *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 16, No. 3, May/June, pp. 395–404.

Doctor, P.G. and Harrington, T.P. (1980) "Analysis of eddy current data using pattern recognition methods", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 137–139.

Engelmore, R.S. and Morgan, A.J. (eds.) (1998) *Blackboard Systems*, Addison-Wesley.

Fu, K.S. (1982) *Syntactic Pattern Recognition and Applications*, Prentice-Hall, Inc.

Georgel, B. and Zorgati, R. (1992) "EXTRACSION: a system for automatic Eddy Currents diagnosis of steam generator tubes in nuclear power plants", *Non-Destructive Testing 92: Proceedings of the 13th World Conference on Nondestructive Testing, Sao Paulo, Brazil, 18-23 October 1992*, C. Hallai and P, Kulcsar (eds.), Vol. 1, pp. 278–282.

Gonzalez, R.C. and Woods, R.E. (1993) *Digital Image Processing*, Addison-Wesley, Reading, MA.

Hartigan, J.A. (1975) *Clustering Algorithms*, John Wiley and Sons, Inc.

den Hartog, J.E. (1995) *A Framework for Knowledge-Based Map Interpretation*, Ph.D. Thesis, Delft University of Technology, The Netherlands.

Hellyar, M.T., Ifeachor, E.C., Mapps, D.J., Allen, E.M., and Hudson, N.R. (1995) "Expert System Approach to Electroencephalogram Signal Processing", *Knowledge-Based Systems*, Vol. 8, No. 4, August, pp. 164–173.

van't Hoff, P. (1996), *Design of a case-base for Eddy Current CBR system: optimal case representation and retrieval*, TNO-report, FSP-RPT-960101.

Hopgood, F.F., Woodcock, N., Hallam, N.J., and Picton, P.D. (1993) "Interpreting ultrasonic images using rules, algorithms and neural networks", *European Journal of NDT*, Vol. 2, No. 4, April, pp. 135–149.

Hu, M.-K. (1962) "Visual Pattern Recognition by Moment Invariants", *IEEE Transactions on Information Theory*, Vol. 8, February, pp. 179–187.

Jarmulak, J. (1997) "A Method of Representing and Comparing Eddy Current Lissajous Patterns", *Review of Progress in Quantitative Nondestructive Evaluation*, Vol. 16A, D.O. Thompson and D.E. Chimenti (eds.), Plenum Publishing, New York, pp. 303–308.

Kamgar-Parsi, B. and Kamgar-Parsi, B. (1990) "Simultaneous fitting of several planes to point sets using neural networks", *Computer Vision, Graphics, and Image Processing*, Vol. 52, pp. 341–359.

Kang, S.J. and Kwon, Y.R. (1995) "Hybrid knowledge-based architecture for building an intelligent nondestructive signal inspection system", *Knowledge-Based Systems*, Vol. 8, No. 1, February, pp. 21–31.

Khotanzad, A. and Lu, J.-H. (1991) "Shape and texture recognition by a neural network", *Artificial Neural Networks and Statistical Pattern Recognition: Old and New Connections*, I.K. Sethi and A.K. Jain (eds.), Elsevier Science Publishers, pp. 109–131.

Kooperberg, C., Bose, S., and Stone, C.J. (1997) "Polychotomous regression", *Journal of the American Statistical Association*, Vol. 92, pp. 117–127.

Krishnapuram, R. and Freg, Ch.-P. (1992) "Fitting an unknown number of lines and planes to image data through compatible cluster merging", *Pattern Recognition*, Vol. 25, No. 4, pp. 385–400.

Krishnapuram, R. and Keller, J.M. (1993) "A possibilistic approach to clustering", *IEEE Transactions on Fuzzy Systems*, Vol. 1, No. 2, May, pp. 98–110.

Krishnapuram, R., Nasraoui, O., and Frigui, H. (1992) "The fuzzy C spherical shells algorithm: A new approach", *IEEE Transactions on Neural Networks*, Vol. 3, No. 5, September, pp. 663–671.

Lacasse, V., Hay, J.R., and Hay, D.R. (1988) "Pattern Recognition of Ultrasonic Signals for Detection of Wall Thinning", *Signal Processing and Pattern Recognition in Nondestructive Evaluation of Materials*, C.H. Chen (ed.), Springer-Verlag, pp.189–198.

Levy, A.J., Oppenlander, J.E., Brudnoy, D.M., Englund, J.M., Loomis, K.C., and Barsky, A.M. (1993) "Dodger: an expert system for eddy current evaluation", *Materials Evaluation*, Vol. 51, No. 1, pp. 34–44.

Lim, T.-S., Loh W.-Y., and Shih, Y.-S. (1997) *An Empirical Comparison of Decision Trees and Other Classification Methods*, Technical Report 979, Department of Statistics, University of Wisconsin, Madison, June 30.

Lord, W. and Satish, S.R. (1984) "Fourier descriptor classification of differential eddy current probe impedance plane trajectories", *Review of Progress in Quantitative Nondestructive Evaluation*, Vol. 3A, D.O. Thompson and D.E. Chimenti (eds.), Plenum Publishing, New York, pp. 589–603.

Lunin, V. and Gomonov, D. (1994) "Neural Network Techniques for Defect Classification in Steam Generator Tube Testing", *6th European Conference on Non Destructive Testing, Nice 94*, pp. 1349–1353.

McNab, A. and Dunlop, I. (1991) "AI Techniques Applied to the Classification of Welding Defects from Automated NDT Data", *British Journal of NDT*, Vol. 33, No. 1, January, pp. 11–18.

McNab, A., Cornwell, I., and Dunlop, I. (1994) "A Framework for Improved NDT Data Interpretation", *INSIGHT*, Vol. 36, No. 5, May, pp. 326–330.

Murtagh, F. (1985) "Multidimensional Clustering Algorithms", *Compstat Lectures 4: Lectures in Computational Statistics*, J.M. Chambers, et al. (eds.), Physica-Verlag, Wuertzburg-Vienna.

Nair, S., Udpa, S., and Udpa, L. (1993) "Radial basis functions network for defect sizing", *Review of Progress in Quantitative Nondestructive Evaluation*, Vol. 12A, D.O. Thompson and D.E. Chimenti (eds.), Plenum Publishing, New York, pp. 819–825.

Oatley, G., Tait, J., and MacIntyre, J. (1998) "A Case-Based Reasoning Tool For Vibration Analysis", *Applications and Innovations in Expert Systems VI: Proceedings of Expert Systems 98, the 18th SGES International Conference on Knowledge Based Systems and Applied Artificial Intelligence, Cambridge, Dec. 98*, R.W. Milne, A.L. Macintosh, and M. Bramer (eds.), Springer-Verlag, London.

Oppenheim, A.V. and Schafer, R.W. (1975) *Digital Signal Processing*, Prentice-Hall, Englewood Cliffs, NJ.

Papadakis, E.P. and Mack, R.T. (1997) "Will Artificial and Human Intelligence Compete in NDT", *Materials Evaluation*, May, pp. 570–572.

Perner, P. (1993) "Case-Based Reasoning for Image Interpretation in Non-Destructive Testing", *EWCBR-93 Proceedings*, Vol. 2, pp. 403–409.

Perner, P. (1996) "Ultra Sonic Image Interpretation for Non-Destructive Testing", *IAPR Workshop on Machine Vision Applications*, Tokyo, Japan, pp. 552–554.

Perner, P. (1998) "Different Learning Strategies in a Case-Based Reasoning System for Image Interpretation", *Advances in Case-Based Reasoning: EWCBR-98 Proceedings*, B. Smyth and P. Cunningham (eds.), Springer-Verlag, pp. 251–261.

Porter, B.W., Bareiss, R., and Holte, R.C. (1990) "Concept Learning and Heuristic Classification in Weak-Theory Domains", *Artificial Intelligence Journal*, Vol. 45, No. 1–2, pp. 229–264.

Pratt, D. and Sansalone, M. (1991) "The Use of a Neural Network for Automatic Impact-Echo Signal Interpretation", *Review of Progress in Quantitative Nondestructive Evaluation*, Vol. 10A, D.O. Thompson and D.E. Chimenti (eds.), Plenum Publishing, New York, pp. 667–674.

Quinlan, J.R. (1993) *C4.5: Programs for Machine Learning*, Morgan Kaufmann Publishers, Inc., San Mateo, CA.

Rajagopalan, C., Raj, B., and Kalyanasundaram, P. (1996) "The role of artificial intelligence in non-destructive testing and evaluation", *INSIGHT*, Vol. 38, No. 2, February, pp. 118–123.

Schaffer, C. (1994) "A Conservation Law for Generalization Performance", *Machine Learning: Proceedings of the Eleventh International Conference (MLC-94)*, pp. 259–265.

Schalkoff, R.J. (1992) *Pattern Recognition: Statistical, Structural and Neural Approaches*, John Wiley & Sons, Inc.

Stepinski, T. (1990) "Analysis of eddy current patterns", *British Journal of NDT*, Vol. 32, No. 12, December, pp. 631–633.

Stepinski, T. (1991) "Analysis of eddy current patterns: Review and recent results", *Review of Progress in Quantitative Nondestructive Evaluation*, Vol. 10A, D.O. Thompson and D.E. Chimenti (eds.), Plenum Publishing, New York, pp. 943–950.

Stepinski, T. and Maszi, N. (1993) "Conjugate spectrum filters for eddy current signal processing", *Materials Evaluation*, Vol. 51, No. 7, July, pp. 839–845.

Udpa, S.S. and Lord, W. (1984), "A Fourier descriptor classification scheme for differential probe signals", *Materials Evaluation*, Vol. 42, August, pp. 1136–1141.

Udpa, S.S. and Lord, W. (1986) "Fourier descriptor classification of magnetite buildup in PWR steam generators", *Review of Progress in Quantitative Nondestructive Evaluation*, Vol. 5A, D.O. Thompson and D.E. Chimenti (eds.), Plenum Publishing, New York, pp. 233–240.

Udpa, S.S. (1988) "Signal Processing for Eddy Current Nondestructive Evaluation", *Signal Processing and Pattern Recognition in Nondestructive Evaluation of Materials*, C.H. Chen (ed.), Springer-Verlag, pp. 129–144.

Udpa, L. and Udpa, S.S. (1990) "Eddy current defect characterization using neural networks", *Materials Evaluation*, Vol. 48, March, pp. 342–347.

Wang, B., Basart, J.P., and Moulder, J.C. (1997) "Fast Eddy Current Forward Models Using Artificial Neural Networks", *Review of Progress in Quantitative Nondestructive Evaluation*, Vol. 16A, D.O. Thompson and D.E. Chimenti (eds.), Plenum Publishing, New York, pp. 217–224.

Wielinga, T.S., Kerckhoffs, E.J.H., and Lorenz M. (1994) "Towards the classification of round and planar weld defects with neural networks", *Neural Network World*, Vol. 4, pp. 465–478.

Wunsch, P. and Laine, A.F. (1995) "Wavelet Descriptors for Multiresolution Recognition of Handprinted Characters", *Pattern Recognition*, Vol. 28, No. 8, August, pp. 1237–1249.

Yan, W. and Upadhyaya, B.R. (1996) "An integrated signal processing and neural networks system for steam generator tubing diagnostics using eddy current inspection", *Ann. Nucl. Energy*, Vol. 23, No. 10, pp. 813–825.

Zahn, Ch.T., and Roskies, R.Z. (1972) "Fourier descriptors for plane closed curves", *IEEE Transactions on Computers*, Vol. 21, No. 3, March, pp. 269–281.

Chapter 6

Application 1: EC heat-exchanger inspection

One test case within the ANDES project concerned developing an automatic interpretation system for the eddy-current inspection of heat exchangers. This chapter describes results of the research done on that problem. The chapter begins with a description of the objects that have to be inspected, the technique used for the inspection, and the inspection procedures. Special attention is paid to the data coming from the inspection system and the way various defects, construction elements, and noise are manifested in that data.

After it is clear what data is available and how it is interpreted by the operators, a consideration is given to the choice of the automatic interpretation technique. The chosen technique – CBR – has been implemented in a prototype software called LISSA. Interesting details of its design are presented. Then, the results of tests on EC data from calibration pieces and data from real inspection are presented and discussed. The chapter ends with a discussion of the perspectives of the deployment of the system.

6.1 Problem description

6.1.1 Inspection of heat exchangers

Heat exchangers. Heat exchangers are used either to increase or decrease temperature of one fluid using another fluid with appropriate (lower or higher) temperature. An example application is a steam generator of a nuclear power plant. Here, the primary-circuit fluid from the nuclear reactor flows through a set of pipes. The transferred heat boils water, creating steam which is used to drive the turbines. Typical modern steam generators contain around 5500 vertical Inconel U-tubes approximately 7.5 m high, and having 15.5 mm internal diameter by 1 mm wall thickness [Bowker et al., 1995]. Another example of heat exchangers are condensers. Condensers used in power stations may have 20 to 30 thousand tubes. The tubes used in these condensers are usually made of brass with a diameter of about 25 mm.

Materials used for heat-exchanger tubes must be resistant to corrosion and heat. Materials often used are Inconel, stainless steel, brass, and aluminium. The diameter of tubes is usually in the range from 15 to 60 mm. The wall thickness is typically between 1 to 3 mm. Apart from tubes, heat exchangers have other construction elements like:

- tube sheets — used to seal the heat exchanger,
- support plates (baffles) — used to support the tubes and usually made of cheap carbon (thus ferromagnetic) steel,
- anti-vibration bars — used to suppress unwanted vibration of the tubes.

Many different defects may be present in heat-exchanger tubes. Typical defects are: cracks, pitting, thinning and dents. Defects may be caused by different processes. Vibration may cause cracks, ring like defects under baffles, or fretting wear. Process fluids or gases may cause corrosion. Contamination in the process fluids or gases may cause erosion. Obstacles in the flow may lead to cavitation, which, in turn, leads to erosion. Various combinations of defects can also occur. The types of defects one looks for during an inspection depend on the type and the use of a heat exchanger.

Inspection methods. Critical heat exchangers are inspected on a regular basis. A number of inspection methods, some of them already mentioned in Chapter 2, are available:

- standard eddy current method (EC),
- magnetically-saturated eddy current method (MSEC),
- remote-field eddy current method (RFEC),
- magnetic flux leakage method (MFL),
- visual inspection with endoscopes,
- ultrasonic methods.

The most widely used inspection technique for heat-exchanger tubes is the standard eddy-current inspection. It can be used for conducting, non-ferromagnetic materials. Ferromagnetic materials can also be inspected using eddy currents but this requires either magnetic saturation or remote field techniques, see Section 2.2.1. Another electromagnetic method used for tube testing is magnetic flux leakage (MFL) testing, but it is usually used for larger-diameter tubing. Endoscopy is used to visually inspect heat exchangers for damage on the inner radius of the tubes. Using a flexible long glass-fibre cable also defects that are remote and difficult to reach can be characterised.

Ultrasonic methods are, in spite of their higher precision, rarely used for bulk tube inspection, the main reason being an overall low speed of ultrasonic inspection. Ultrasonic tube inspection requires use of either a transducer array or rotating transducers. Especially, when rotation transducers are used the pull-speed of the probe has to be low. Additional inconvenience when doing ultrasonic inspection is the need to fill the tubes with a liquid. In practice, ultrasonic techniques are only used for checking of the results of EC testing, for example, to check suspicious spots under baffles or in the proximity of tube sheets.

6.1.2 Eddy-current inspection

Most eddy-current inspections are performed according to some standard procedure, which can be an internal procedure or a recognised international procedure [ASME, 1995; Poetz et al., 1992]. A typical inspection procedure comprises the following actions:

- choice of probe type,
- choice of inspection frequency(ies),
- calibration on samples with known defects,
- measurement on heat exchanger,
- on- or off-line interpretation of data,
- generation of a report.

Choice of probe type. Eddy current inspection can be done either with a single-coil absolute probe or with a differential probe which uses two coils. In absolute mode the impedance of the single coil is compared to a reference coil with fixed impedance. Use of an absolute probe has the disadvantage that any gradual changes in the specimen parameters or in the parameters of the coil (e.g., due to temperature) will cause the signal to drift as the probe is scanned. The differential probe does not exhibit this disadvantage. In such a probe, two coils are placed next to each other and the values of impedance of the two coils are compared during the scan. In this way both coils will see the same gradual changes and their influence will be cancelled. However, any abrupt changes in the specimen will be detected by the differential probe.

In practice, the absolute probe is used when long gradual defects are expected (such as erosion) which could be missed by a differential probe. The differential probe is, for example, good at detecting pitting.

The most commonly used probe has two bobbin coils and provides both absolute and differential signals simultaneously. Its main advantage is the speed at which the measurements can be made — the probe can be pulled at speeds of several decimetres per second. A disadvantage of the bob-

bin probe is that it cannot detect purely circumferential small defects (a defect has to have some axial extent to disturb the eddy currents).

Another probe type used for EC inspection is a rotating pancake probe. It uses a small diameter coil whose axis is perpendicular to the tube surface. The coil is sprung to the tube surface and rotated around the circumference using a motor. It can detect defects at any orientation and it also gives information about the circumferential location of the defects. As the probe is pulled through the tube the coil describes a helix. In order to give a full tube coverage the probe has to be pulled much slower than a bobbin probe — a typical rotating probe is pulled at a few millimetres per second.

Another probe type is an array probe built of several pancake coils around the circumference. It combines the advantages of the bobbin and the rotating probes, but requires special equipment which can operate several coils at once.

When eddy-current measurements are made only with absolute probes, the differential signal can be obtained by differentiating the absolute signal. So the question here is more what signals will be displayed (and recorded) on the instrument. The advantages and disadvantages of absolute and differential probes are summarised in Table 6.1.

Table 6.1: Comparison of absolute and differential probes (from [Cecco et al. 1983]).

Advantages	Disadvantages
Absolute probes	
• respond to both sudden and gradual changes in properties and dimensions • combined signals are usually easy to separate (simple interpretation) • show total length of defects	• prone to drift from temperature instability • more sensitive to probe wobble than a differential probe
Differential probes	
• not sensitive to gradual changes in properties or dimensions • immune to drift from temperature changes • less sensitive to probe wobble than an absolute probe	• not sensitive to gradual changes (may miss long gradual defects entirely) • will only detect ends of long defects • may yield signals difficult to interpret

Choice of inspection frequency. The frequency is chosen depending on the type (especially thickness) of the tube and on the expected location of the defects (inside/outside) and their type. The choice is made based on the dependence of the skin depth on the frequency as described in Section 2.2.1. If a multiple-frequency scan is to be carried out, frequencies are chosen that give the best mixing results, the best separation of artefacts from defects, etc.

Calibration on sample with known defects. Before each inspection the inspection equipment is set up and calibrated based on scans of a calibration pipe or pipes (see Figure 6.1 & Figure 6.3). A calibration pipe contains machined defects of the same type as expected during inspection. The equipment is set up so that it can detect all the calibration defects and can also distinguish between various defect types. System settings that can be adjusted based on the calibration scans include the frequency, gain and phase. In some cases, the calibration pipe can guide the choice of the right probe for the inspection, for instance, as far as the probe resolution is concerned, see, for example, [Pfisterer, 1985].

The most important use of the calibration pipe is to construct a calibration curve. A calibration curve defines the wall loss (in %) due to a defect, as well as the defect location (outside or inside defect – ID or OD), in relation to the phase angle of the defect signal. A calibration curve is con-

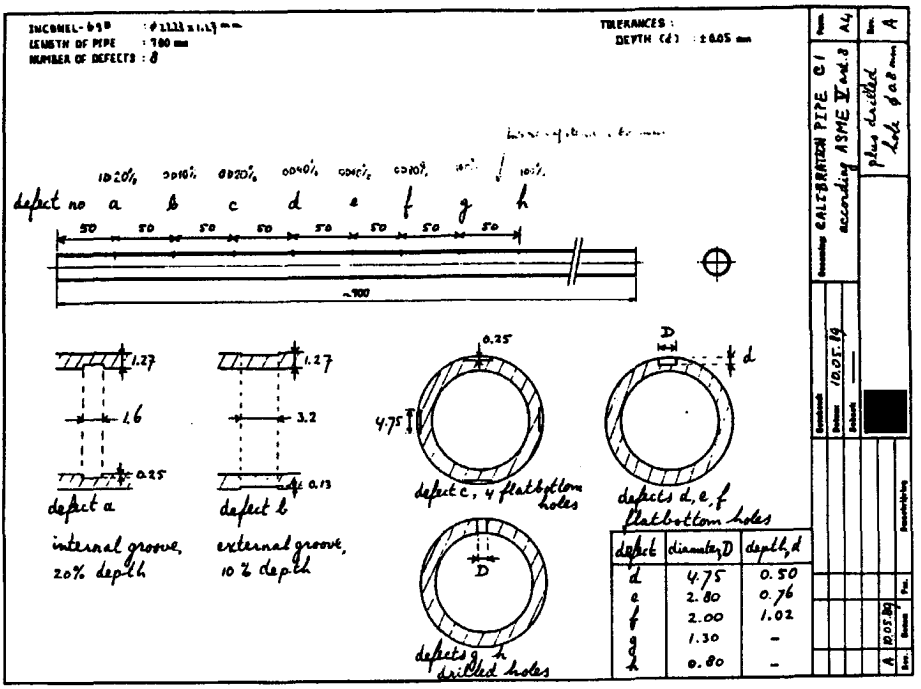


Figure 6.1: Drawing of an ASME calibration pipe.

structed by scanning the calibration pipe and for every detected defect calculating the phase of the signal and specifying the corresponding wall loss and defect location (ID/OD). Through the so obtained data set two curves are fitted (usually parabolas) which define the ID and OD portions of the calibration curve. In modern digital inspection systems construction of a calibration curve is largely automated. Figure 6.2 shows an example of a calibration curve.

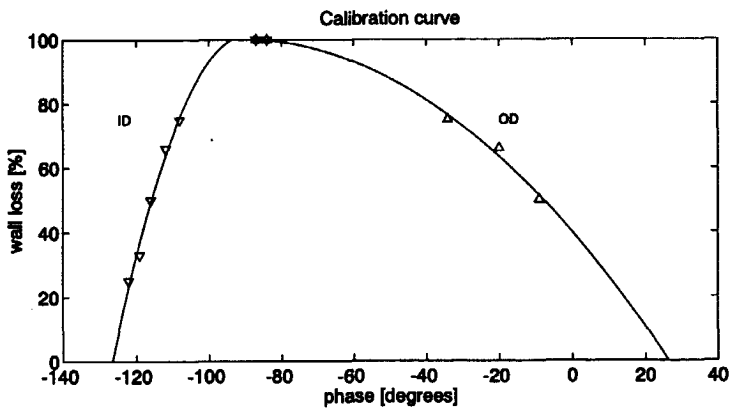


Figure 6.2: Example of a EC calibration curve. Two parabolas are fitted through the data – one for internal defects (ID) and one for outside defects (OD).

Sometimes, a calibration is done on the amplitude of the absolute signal. This is used if the wall loss of the tube occurs over longer distances and has been caused by uniform mechanism (e.g., erosion) over the whole tube. (This method it is not able to determine ID/OD location of the wall loss.)

The calibration phase of the inspection procedure can also be used to determine the mixing parameters. Modern, computer-based systems can automatically calculate them from a signal from a calibration baffle plate.

Measurement on heat exchanger and on-line interpretation of data. Usually, heat exchanger scanning is done by a team of at least two persons. One person pushes the cable with the probe to the end of the tube and then pulls it back. This can be done either by hand or using a push-puller. When the probe is being pulled, the signal can be observed on a display of the scanning equipment. When no automatic defect detection is used then the signal has to be interpreted directly as it appears on the screen. If data is not being recorded then, when a (possible) defect is detected, often the probe is pulled a second time slower along the defect to facilitate signal interpretation.

If automatic detection is used then, usually, the signal from the whole pipe is first acquired. The system detects possible defects and determines their wall-loss using the calibration curve. Afterwards, the operator checks the defects found by the system. Meanwhile, the probe is moved to another tube. More sophisticated systems can store either the whole data, or the biggest defects, indexing them by the pipe coordinates. This information can later be used to construct a report from the inspection.

Off-line interpretation of data. In modern systems, all the signals are digitised and stored. This makes it possible to apply advanced signal processing either off-line in the field or later in a lab. This can be useful if, for example, a non-optimal calibration was done. If all the data has been recorded than it can be reanalysed in the lab.

Generation of report. In the past inspection reports consisted of a listing of the pipes with the most serious defects and were made by hand. Sometimes, (selected) signals were recorded on strip charts and included with the report. In a report, pipes are usually identified by their row and column coordinates.

Computer-based equipment makes it possible to generate inspection reports almost automatically. Reports may include graphical representations of the heat-exchanger with the tubes colour coded depending on the seriousness of the defects.

The way the signal is interpreted and reported depends on the type of damage. For example, when a lot of corrosion is present then only the overall corrosion level and several biggest defects per pipe are reported. The number of defects present and their position is not reported. In a not corroded pipe where only cracks and baffle damage are present, one may choose to report all the detected defects.

Because of a large number of tubes in a heat exchanger, inspection may take much time. Bowker et al. [1995] describe an inspection of four steam generators (22504 tubes) which took 23 days to complete, from an arrival on the site to submitting a preliminary report and leaving the site. According to the information obtained from TMT–Kontroll Technik, an inspection of a condenser in a power plant having 20-30 thousand tubes takes 10-14 days.

6.1.3 Eddy-current data from heat exchangers — typical signal responses

An eddy-current signal may contain various types of indications. Four categories of indications are listed in Table 6.2. Two or more indications can be located so close to one another that it is impossible to separate them (e.g., multiple-defects, defects under baffles). The seriousness of such defects

can be difficult to determine, and may require a use of multifrequency EC inspection or even a use of another NDT inspection method (e.g., IRIS — Internal Rotating Inspection System by ultrasound).

Table 6.2: Categories of eddy-current indications from heat-exchanger inspection.

Indication category	Examples
defects	cracks, pitting, cavitation damage, ring or half-ring damage, long uniform damage
non-defects, artefacts	dents, magnetic inclusions, conducting and magnetite deposits, weld inhomogeneity
construction elements of the heat exchanger	baffle plates, anti-vibration bars, tube sheets,
noise and other signal distortions	measuring equipment noise, pilgrim noise, weld, corrosion, probe wobble, drift, off-set, pipe entry/exit

Indications from defects. Below is a list of the most typical damage types occurring in heat exchangers:

- *Pitting* — Pitting damage has a form of small pits. (On calibration pipes it is usually simulated as flat-bottom or round-bottom holes, see Figure 6.3.) Pitting is mainly caused by corrosion and is very common. If pitting is expected then the inspection is done with a differential probe. Indications in the signal have shape of a flattened figure "8" (see Figure 6.4). Depth of the pits can be determined based on the phase of the indication by using a calibration curve. However, this is less precise if the actual pitting has a different diameter then the holes used for the calibration. Pitting can be very dense resulting in a superposition of overlapping indications.
- *Cracks* — Cracks result in similar indications as pitting. The detectability of cracks depends on their orientation and the type of probe used. When a bobbin coil is used then axial cracks can usually be detected without problems, however, sizing of such defects can be difficult. Detection of circumferential cracks using a bobbin probe is difficult, because then the crack is parallel to the eddy currents in the tube. To guarantee detection of such cracks a different coil type should be used, for instance, a rotating or array probe.

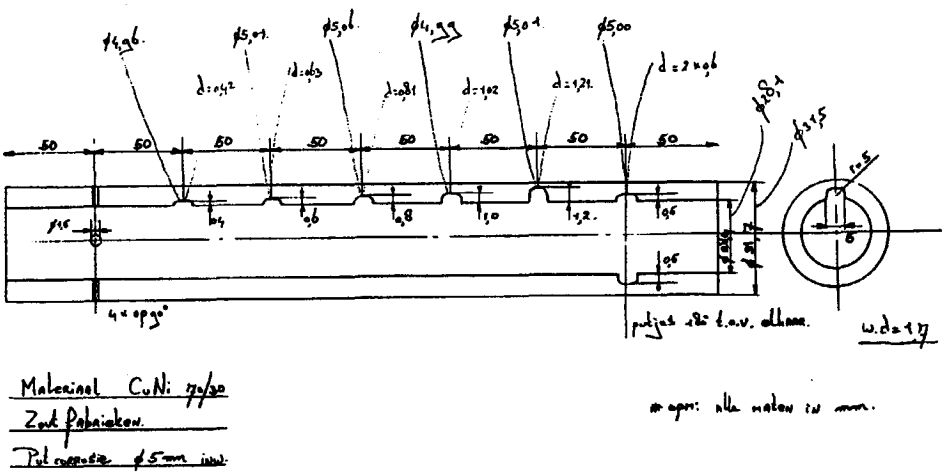


Figure 6.3: Drawing of a calibration pipe for pitting.

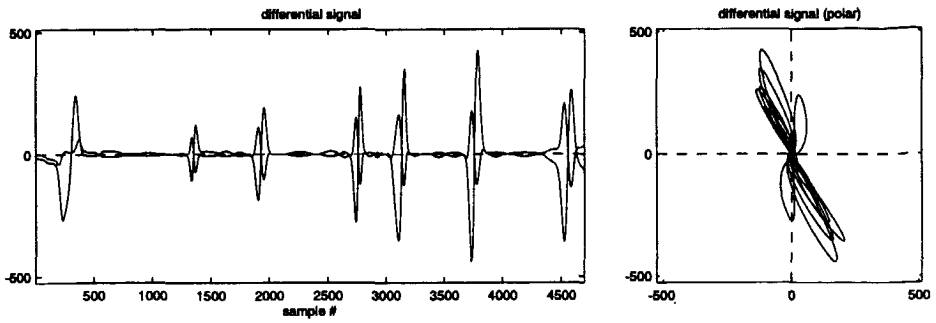


Figure 6.4: Typical differential signal response to pitting defects — flat figure "8" signal.

- *Cavitation damage* — Presence of some obtrusion (e.g., sea shells in a sea-water cooled condenser) in a pipe may cause cavitation damage. The form of damage can be quite complex, thus resulting in complex signal indications from a differential probe. Still, a rough estimation of the depth can be given based on the phase of the signal.
- *Ring and half-ring damage* — This damage occurs on the outside of a pipe. It is usually caused by the vibration of the pipe in a baffle. Depending on the kind of vibration, the damage can cover the whole or only a part of the pipe circumference. A half-ring damage can also be caused by, for example, corrosive agents dripping on the pipe. The axial length of the ring damage is in the order of several millimetres, and usually a differential probe is used to inspect it. Depending on the axial length of the damage the form of the differential signal will vary from an 8-like to an S-like shape, like in Figure 6.5. The indications from the baffle damage will of course be influenced by the signal from the baffle.

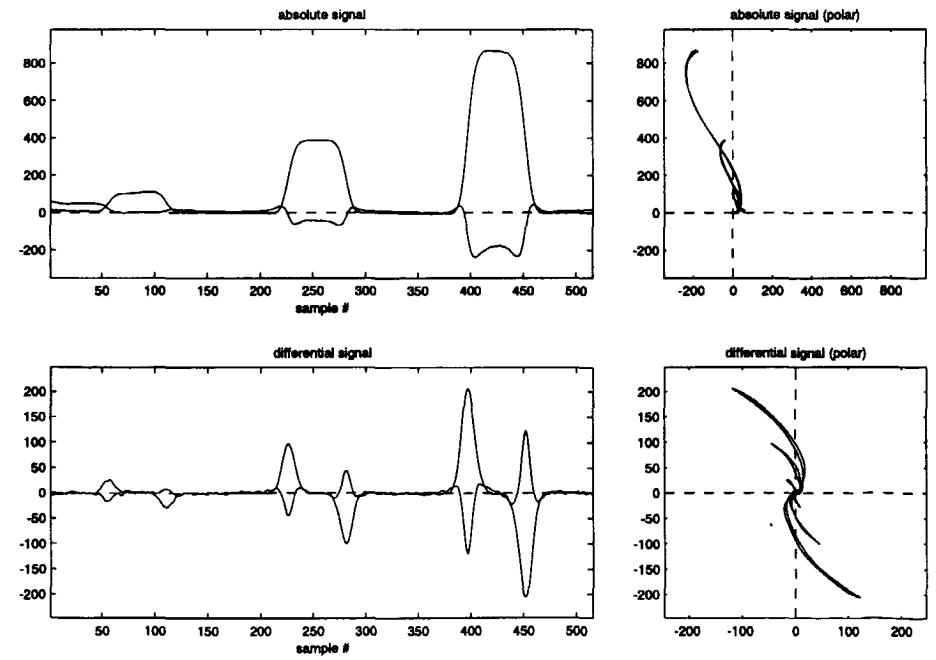


Figure 6.5: S-shaped response from wide, sharp-edged grooves.

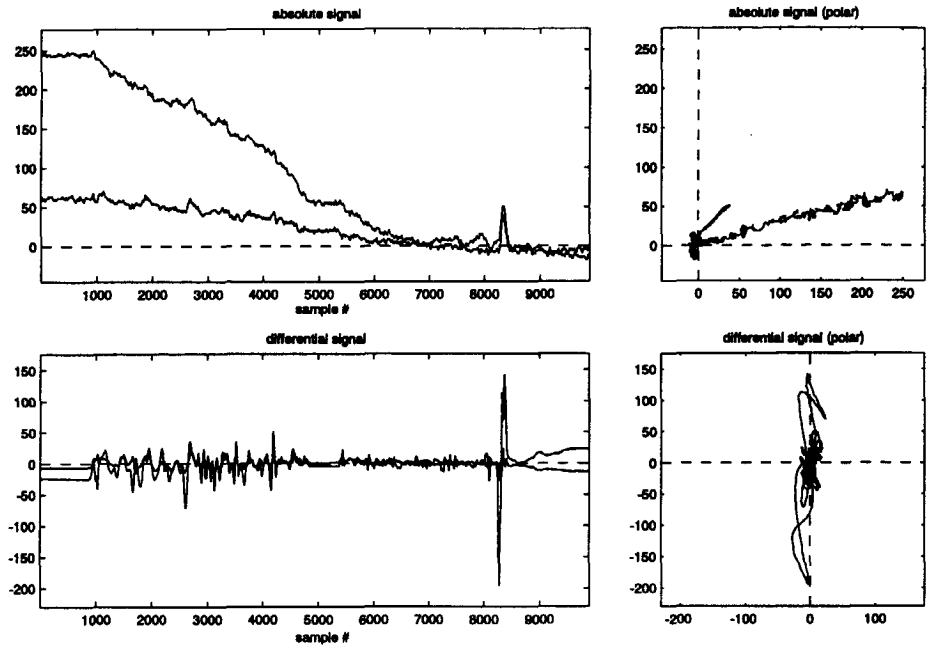


Figure 6.6: Signal from fretting wear damage (the peak signal at sample #8600 is from a hole outside of the fretting wear area).

- *Long, uniform damage* — This can be caused, for instance, by corrosion, erosion, or fretting wear. Corrosion and erosion result in a more or less uniform wall loss over the circumference. Fretting wear (caused by tubes vibrating against each other) has usually a form of large flat spots on the outside of the tube. This type of defect is almost undetectable using differential probes because of smooth edges, therefore, if it is expected then an absolute probe is used for inspection (see Figure 6.6). If the type of damage is known, for example if there is only flat damage, then it can be sized using amplitude of the absolute signal. Complex damage of unknown type may require a use of a rotating probe to do accurate sizing.

Indications from non-defects and artefacts in the tube. Apart from indications from defects, the signal may contain, largely similar to them in form, indications from artefacts in the tube and other damage which is not considered as defect:

- *Dents* — Usually, dents are not considered defects as long as the thickness of the wall has remained unchanged. Dent indications usually look like very flat "8" shapes (almost straight lines), and their phase angle is outside the calibration curve range. (For example, Figure 6.21 has a signal from two small dents.)
- *Magnetic inclusions, conducting and magnetite deposits* — Magnetic inclusions can be introduced into a non-ferromagnetic tube during manufacturing process. The cooling or heating medium may contain conducting (usually copper) or magnetite elements which may deposit on the tube. All these yield very strong EC signals. At certain frequencies they could be mistaken for defect signals; however, when the frequency is changed the signals from magnetic and conducting artefacts will rotate differently than signals from defects. Thus, usually, multifrequency inspection will be capable to distinguish these artefacts from defects. Still, deposits occur often near the construction elements (baffles, tube sheets) yielding a difficult to analyse signal.

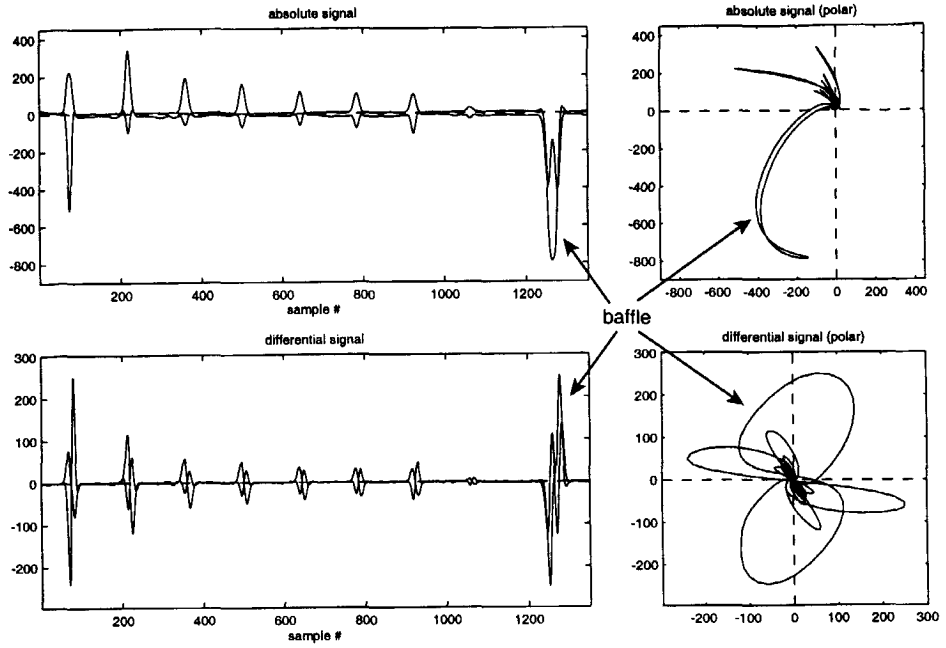


Figure 6.7: Signal from pitting and from a baffle.

Indications from construction elements of heat exchanger. The elements of a heat-exchanger construction may also be visible in the signal, often as indications having large amplitude:

- *Baffle plates* — The form of baffle-plate indications depends on the material of the baffle plate, baffle-plate thickness, tube thickness, and test-signal frequency. Often, the differential baffle-plate indication looks like a fat figure "8" shape (see Figure 6.7), and the phase of the indication lies outside the calibration curve range. However, it is also possible that the baffle indications look like defect indications and lie within the same phase-angle range. In such case, multifrequency inspection is necessary to reliably distinguish baffles. While the phase of defect indications in all frequencies corresponds to (approximately) the same wall loss, the phase of the baffle indications will correspond to different wall-loss values in each frequency used. Baffles often occur in combination with defects.
- *Anti-vibration bars* — Anti-vibration bars give indications similar to those from baffles, however, the signal amplitude is smaller because the bars do not encircle the tube.
- *Tube sheets* — Tube sheets of a heat exchanger give very strong indications. Usually, one is interested not in the tube sheets but in any defects close to them (e.g., cracks due to corrosion under deposits). It is difficult to detect defects under a tube sheet with the same system settings as used for inspection of the rest of the tube. If damage under a tube sheet is expected then special settings have to be used which expose the defects.

Indications from noise sources. All indications that are not defects, which usually cover long sections of the tube, and which make signal interpretation more difficult are considered here as noise:

- *Measuring equipment noise* — This noise has a broad frequency spectrum and it actually looks like noise, see Figure 6.8. It is usually a problem only when the tube itself is clean and the defects are very small, for example cracks, and difficult to detect within the noise. Fortunately, various filtering techniques can often be successfully used to remove this type of noise.

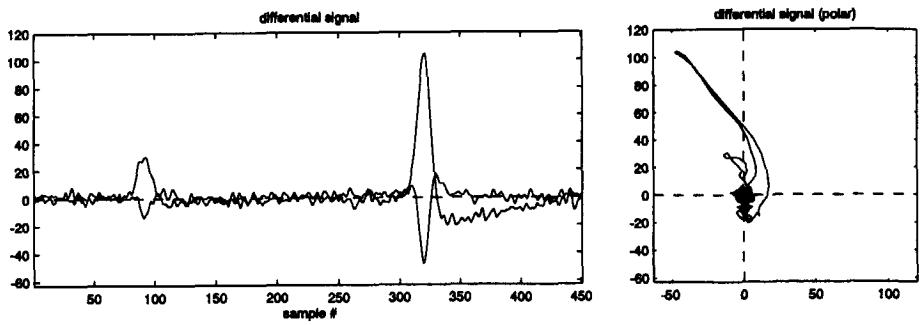


Figure 6.8: Measurement noise in the signal.

- *Pilgrim noise* — This is caused by the internal pipe diameter variations due to the pipe manufacturing process. The pilgrim noise has higher amplitude than equipment noise and can distort the defect signals. However, this type of noise is highly periodic and algorithms can be devised to remove it if the signal has been sampled with a position encoder (simple frequency filtering will usually not work because the defect indications have roughly the same frequency content as the pilgrim noise).
- *Corrosion* — The shallow corrosion (10–20% wall loss) indications look like oscillations along one direction, see Figure 6.9. Usually, the overall amount of corrosion is reported but the individual corrosion pits are not sized, except for pits having significant depth (if such are present). The small corrosion signal can be ignored by setting detection thresholds appropriately or by doing deepest-first detection, see Section 6.4.3.
- *Probe wobble* — Non-axial movement (wobble) of the probe in the pipe is visible in the signal as oscillation along one direction. These indications can sometimes be ignored by setting appropriate detection threshold; however, the wobble can distort signals of small defects. In such cases the wobble signal could be removed, but this is not trivial. Benoist et al. [1995] mention an adaptive filtering method which can remove wobble indications.
- *Offset and drift* — Ideally the differential signal should be centred around zero. When the system settings are incorrect then the whole differential signal can be offset from zero by a constant value. Drift (varying offset, see Figure 6.10) of the differential signal can be caused by, for instance, a varying angle of the probe in the pipe. Both offset and drift can be quite successfully removed from the differential signal. Removal of offset and drift from the absolute signal should be done with caution, as non-zero signal can be a result of real damage in the pipe.

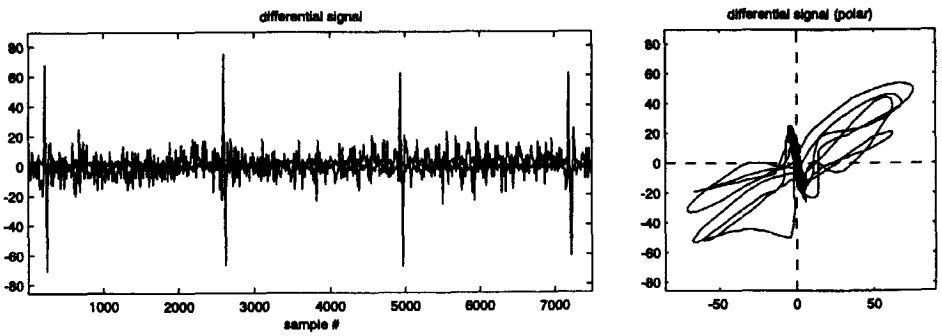


Figure 6.9: Signal from corroded pipe with baffles. All corrosion indications have roughly the same phase.

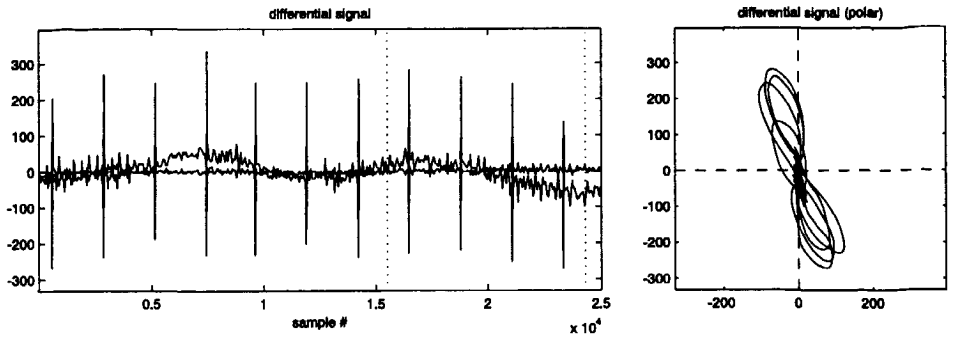


Figure 6.10: Drift in the differential signal.

- *Pipe entry and exit* — Where the probe enters or exits the pipe a jump of high amplitude in the signal is observed. These indications can be ignored. (This has to be distinguished from the tube sheet indications.)

Combinations of indications. Interpretation of indications is made more difficult if they come from a combination of the above described sources. The most often occurring combinations are:

- *Multiple defects* — Due to a limited resolution of a probe, multiple defects close to one another may give a single indication. Figure 6.11 illustrates this on a case of three holes. Usually, these indications are treated as a single defect and only one wall-loss value is determined.

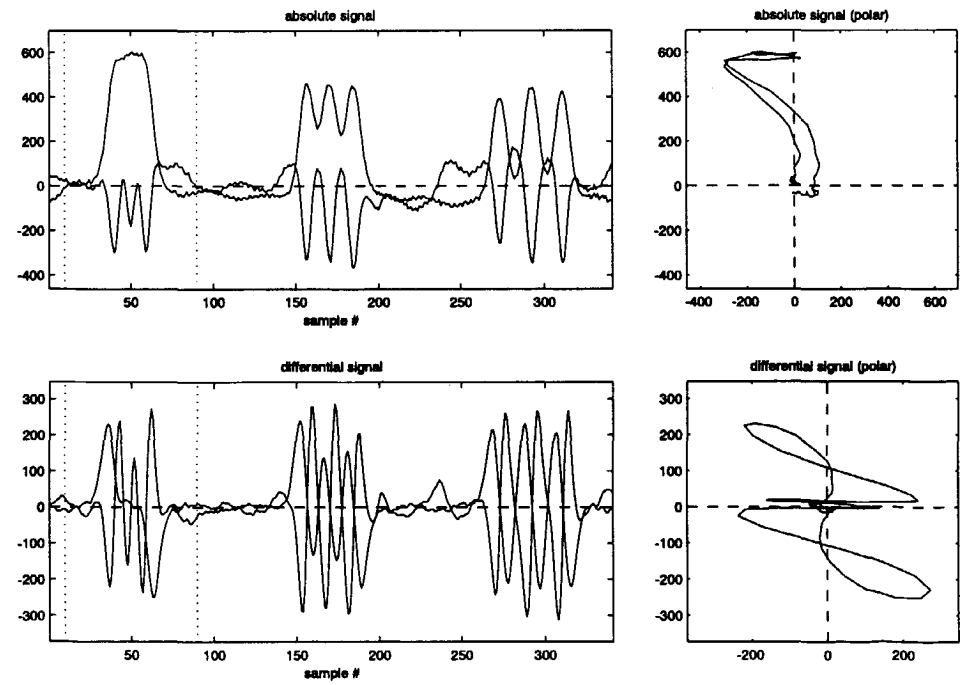


Figure 6.11: Signal from 3 groups of 3 through-wall 1mm diameter holes located 4, 6, and 8 mm from one another. The polar plot shows the first group (4 mm mutual distance).

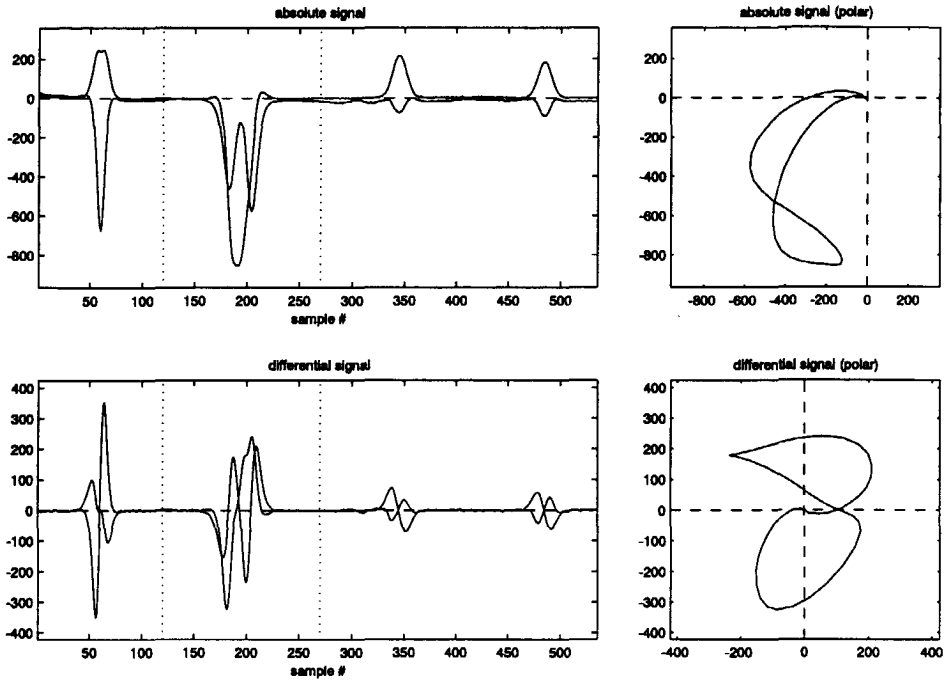


Figure 6.12: Signal from a defect under a baffle among other defect signals on ASME calibration pipe. Compare with Figure 6.7.

- *Defects under baffles* — Figure 6.12 shows an example of an indication coming from a defect under a baffle (compare to a baffle-only signal in Figure 6.7). Usually, the baffle signal can be mixed out and the depth of the defect can be determined from the mix signal.
- *Defects and dents* — Usually, some time after a dent occurred, the wall in that spot will get thinned. The wall loss can be determined by looking at multiple frequencies (with or without mixing).
- *Local defect combined with longer wall-loss defect* — An example of this is pitting combined with erosion. This type of damage is best interpreted looking simultaneously at absolute and differential channels.

6.1.4 Data interpretation: problems, reasons for automation

A number of problems are encountered during interpretation of measurements. One important problem is a considerable workload of the operator, who has to verify each measurement on the screen in a short amount of time. With a total number of pipes measured in the order of several hundreds per day, an overload of the operator and loss of concentration becomes an issue. One of the aims of automated interpretation systems is to reduce the load on the operator, thus preventing errors and increasing reliability.

When the number of defects is large, usually, only the largest defects are reported, because otherwise it would be impossible to keep up with the schedule. This makes it difficult to monitor the damage progress, even if the inspection is done regularly, because no information about the growth of defects is available. Also, most of the systems have no provisions to specify defect type in addition to the percentage of wall loss. It would be advantageous if an automatic system could help in obtaining a more detailed report on the state of the inspected heat exchanger.

The speed of inspection is largely determined by the time to do the mechanical scan of the tubes, and not the interpretation time, therefore, it is not expected that an automatic system could significantly reduce the inspection time. However it can speed up preparing of inspection reports, as well as reduce time necessary for preparing future inspections of the same heat exchanger.

In certain situations interpretation of EC data can become difficult, and the use of computer systems could possibly help the operator in making correct interpretations. A few examples of such inspection problems are:

- The diameter of the pitting present in the pipe does not correspond to the diameter of the pitting in the calibration pipe(s). If the real diameter is smaller than the diameter in the calibration pipes then the wall-loss will be overestimated.
- The pitting present in the pipe has an irregular (for example a horseshoe) form. The differential signal is distorted and difficult to interpret.
- There is so much pitting in the pipe that it is impossible to determine the wall loss for every pit. Still, it is desired to have an indication of the total amount of pitting in the pipe. (Here it is assumed that it is possible to distinguish separate pits.)
- Several defects overlap, which makes the interpretation difficult and the use of calibration curves not reliable.

Some of these data characterisation problems could probably be solved by using an automatic system to analyse the data in more than two frequencies and by using multiparameter calibration curves. However, because of the lack of resources (for preparation of calibration pieces, doing numerical simulations, etc.) these possibilities have not been further studied within the ANDES project. The goal of the project lied in developing efficient and reliable software for supporting an operator in signal interpretation, building upon the interpretation methods currently used by the operators. It was desired that apart from reporting the wall loss, the software should also facilitate reporting of the damage classes. Increasing the consistence of the interpretation of the data (reducing its operator dependence) was also an important goal.

6.2 Choice of AI technique

Before developing a new system, a choice of the technique had to be made. At first, the techniques reported in the NDT literature were considered. In Section 5.4 a number of existing systems for EC data interpretation have been mentioned. Systems presented there use two AI techniques — ANNs and expert systems (some systems also make use of elements of fuzzy set theory and/or could handle uncertainty). However, it turns out that our inspection problem is significantly different from the problems for which the systems described in Chapter 5 have been built. The difference lies mainly in the large variety of inspections that the system would have to handle. The most important implication of this fact is that, before the actual inspection takes place, the knowledge about the heat exchanger to be inspected is limited. Of course, based on the previous inspections one has some idea about the data to be expected, however, this knowledge is not complete. This makes it difficult to use an expert system for interpretation of data. In the already presented (Section 5.4.2) examples of the use of expert systems for EC inspection it can be noticed that they were used for interpretation of data from nuclear power plants — a well-documented inspection problem in a strictly controlled environment.

Use of ANNs (or other classifiers) requires availability of a training set. In our case, the only set that could be available was the calibration data or the data from previous inspections of the same heat exchanger. The problem is that a heat exchanger may contain more and different defects than those in the calibration, and the settings of the system may be different from those used in the previous inspections making use of the old data not reliable. As already explained in Section 5.3.4, ANN classifiers can be unreliable when confronted with unknown data, which is why this methodology was not chosen for our system.

The choice fell on CBR. CBR makes use of a combination of a-priori knowledge about the inspection as well as knowledge contained in the available data. The calibration data can be used to build an initial case-base, and to classify new data. As opposed to ANN, CBR is good at recognising previously unknown data, thus it is reliable, and, what is perhaps most important, CBR systems can learn from the way the operator classifies new data.

6.3 LISSA — introduction

This section presents the design of LISSA, a system for EC data interpretation. The description begins with general design specifications and then gives an overview of the system. Next, parts of the system which fulfil the general functionality of EC inspection software are presented, followed by a description of the CBR part of the system.

6.3.1 Design specifications

The system has to process eddy-current data and do automatic interpretation of the detected anomalies. Of course, it has to be able to perform the tasks normally expected from EC inspection software, this includes the following:

- reading data from the measurements,
- filtering and other preprocessing of data,
- displaying data on a screen in various forms,
- calculation of mixing parameters and doing signal mixing,
- detecting (possible) defects in the signal,
- calculating the phase of a defect,
- constructing a calibration curve,
- wall-loss determination using a calibration curve and a signal phase,
- report generation.

Because only a prototype system has to be developed, it is sufficient that the system can work with data stored in files. A system for a field use would have to handle real-time data from the acquisition hardware. In the prototype version of the system, little attention is paid to the design of the user interface. The priority is to have an interface which enables flexible analysis of the signals. When a field version of the software is to be developed, a better user interface (easier and faster in use) will have to be designed and implemented.

6.3.2 Outline of the system

The overall schema of the system is shown in Figure 6.13. The first three steps: preprocessing, defect detection, and mixing can be found in most eddy-current inspection software. The next step is new. First of all, a case-base of previously seen indications is maintained. It contains cases consisting each of the indication signal, some other parameters used to determine how the matching is done, and the action to be taken for similar indications.

When, during an inspection scan, a possible defect indication is detected, the case-base is searched for a case that contains the most similar indication. As can be seen in Figure 6.13, the search can result in three different situations:

1. A good match is found. The value of similarity measure is better than some predefined threshold $T1$ (the "do-it" threshold, see Section 4.9.2). In such a situation the action defined in the retrieved case will be taken automatically. The new indication will not be stored in the case-base.

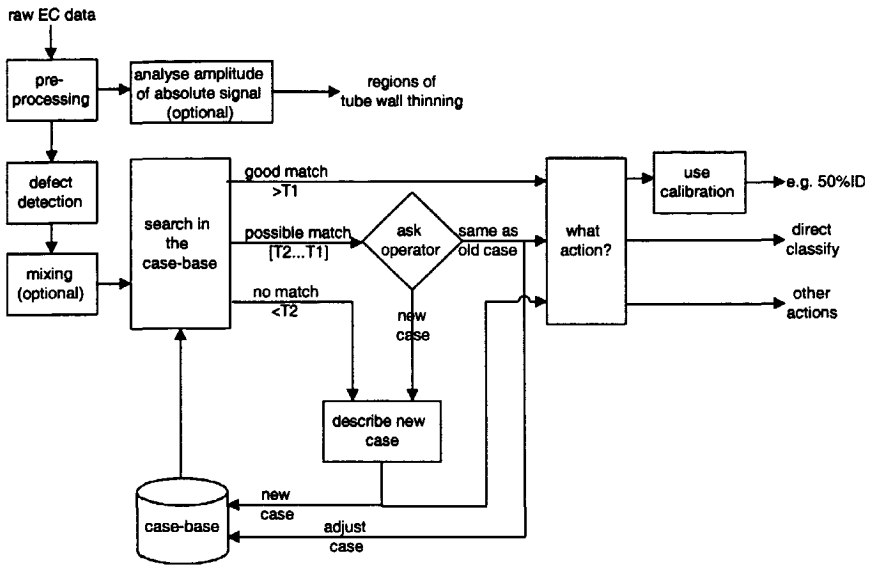


Figure 6.13: Overall schema of LISSA.

2. If similarity is lower than threshold *T1* but still higher than some threshold *T2* (the "tell-me" threshold) then the operator will be asked to confirm if the signal is of the same type as the retrieved case. If so, then action defined for the case will be taken (as in point 1). If not, then the new case will have to be described (as in 3). Anyway, the new case will be stored in the case-base.
3. If the similarity is lower then threshold *T2* then the operator has to give a full description of the new case (including the actions to be taken).

6.4 LISSA — general EC software functionality

6.4.1 Measurement data and system channels

The prototype system can read data from ASCII or binary files. When a file is being loaded, the channels in the file are mapped to the internal channels according to a mapping defined by the user. Internally, the system can work with absolute and differential channel types. Per type, there can be two original channels and one mix channel. To correctly display the length of the tube (when sampled with a position encoder) the system has to be told the value of the intersample distance.

6.4.2 Preprocessing

Preprocessing of the signal facilitates later interpretation. The goals and working of the pre-processing stages implemented in LISSA are described below.

Differentiating the absolute signal. Usually, the system receives the differential signal from the measuring equipment. If only absolute measurements are available, the differential signal can be obtained by differentiation of the absolute signal. To simulate the results that would come from a real differential probe with two coils, the differentiation is done by a convolution of the absolute signal with a response signal. The response used is a derivative of the Gaussian. To obtain realistic results the used response length should be related to the distance between the two coils in the differential probe. This method of obtaining the differential signal will work correctly if the distances between

the sample points are constant (position-triggered sampling). It will not be reliable for time-sampled data.

Filtering of the signal. Three filter types are implemented in the system: a lowpass, a bandstop, and a median filter. These filters can be used to remove at least some noise from the signal. This can improve later defect detection (because threshold crossings are more clear) and case matching (because irrelevant differences between indications are removed). Still, some noise types cannot be removed using these filters without distorting signal from defects. More efficient noise removal procedures are possible, for example, as described in [Benoist et al., 1995], but they have been judged as too complex to implement within the framework of this project.

- *Lowpass and bandstop filters* — The lowpass filter attenuates signal components above certain frequency, while the bandstop filter attenuates signal components within certain frequency range. The filtering is done by convoluting a filter response with the data. Of the two filters the lowpass filter has turned out to be useful to remove high-frequency noise. The bandstop filter was implemented so that its usefulness for, for example, pilgrim noise removal could be tried, however, it does not work well because the noise and the defect signal frequency bands overlap. One has to be careful when using these filters because apart from removing the noise they can distort the signal from defects if used with wrong parameter settings.
- *Median filter* — A median filter replaces signal value at a given position by a median of the signal in the neighbourhood of that position. To make the result of filtering smoother the output of the filter is a combination of the original signal value at a given point and of the calculated median. There are two parameters controlling this filter: the length of the window over which the median is calculated, and the strength of the filter (the fraction of the median in the output of the filter). This filter can be used to remove severe noise from the absolute signal of long defects, and in some situations it performs better than the lowpass filter.

Offset removal. Signal coming from the measuring equipment may have an offset due to the incorrect zeroing (balancing) of the measurement circuits. An offset (relative to the origin) in the differential (and sometimes absolute) signal may negatively influence working of the system (e.g., defect detection), therefore, it has to be removed, *unless* the offset is caused by a wall loss that is to be detected. If a user wants to remove the offset he can do it in two ways:

- *Simple signal balancing at a given point* — This requires only subtracting a value of the signal at the given point from the whole of the signal.
- *Fully automatic method* — An offset in the *differential* signal can be automatically removed by centring the signal on its median value. (Of course, offset removal in the differential signal is not needed when the differential signal has been calculated from the absolute signal by differentiation.) The method used for the *absolute* signal works as follows:
  - first, two histograms (for real and imaginary part of the signal) of the data are made,
  - next, an average of the bin with the most hits is found (for each histogram),
  - finally, the value of the so found maximum is subtracted from the signal.

Removal of the end-distortions from the signal. The signal from the probe entering or leaving a pipe has usually very high amplitude. Often, it is clipped by the AD converter so that no reliable signal interpretation can be done. Unless defects at the tube sheet are to be detected then the automatic signal analysis is simplified when these end-distortions are removed. Also, when the distortions are removed, the signal can be easily scaled on the display for the convenience of the operator.

The end-distortion removal is done in several iterations. Each iteration consists of the following:

- remove distortion from one data end,
- remove signal offset,
- remove distortion from another data end,
- remove signal offset.

Iterations are repeated until no distortion on either data end can be detected. A distortion from one data end is removed by simultaneously scanning all the signals from the end until the absolute values of all the signals fall below user defined thresholds. Thresholds are defined separately for each of the channels and, usually, they lie below the defect detection threshold.

Drift removal. Occasionally, the differential signal has a drift (see Figure 6.10) caused, for instance, by changes in the elements of the measuring circuit or by probe having a skewed stand in a too wide tube. Usually, it is possible to remove the drift in the differential signal reliably. A drift in the absolute signal can be easily confused with a signal from a long defect (see Figure 6.6), therefore, it would be difficult to remove it reliably, and its removal is not implemented in LISSA.

To remove the drift first a signal is calculated which represents the windowed-median of the differential signal, then the so-obtained signal is subtracted from the original signal. Because of the need to repeatedly calculate a median over a long data window drift removal is computationally intensive.

6.4.3 Detection

Before a defect or another event in the signal can be classified, it has to be detected. Generally, defects will correspond to these portions of the signal whose amplitude is above some threshold. When there are few defects in the pipe, they are far from one another, and noise level is low then defect detection is trivial. Such a situation occurs, for example, in calibration pipes. When there are many defects and the noise level is high then defect detection becomes more complicated, especially the determination of the extent of a defect.

The detection process consists of two steps:

- detect the presence of a defect,
- determine the extent of the defect.

Several methods of defect presence and extent determination have been implemented in LISSA, all of them use information about peaks in the signal. Thus, before detection is started, peaks of the signal have to be found.

Peak detection. Peaks are defined as maxima of the absolute value of the signal above some threshold. Two thresholds are used; therefore, two types of peaks are distinguished. The big ones, above the higher threshold, indicate presence of defects. Small ones, above lower threshold, are needed to properly detect small defects with asymmetric amplitude; they are also used for the determination of the extent of a defect. The values of the two thresholds are specified by the user. The peaks are detected once after the signal is loaded and the information about peak positions is stored.

Though peak detection may at first seem to be simple, it is in fact quite difficult, for example, it can be quite problematic for "fat" circular signals for which the peak position is difficult to determine unequivocally, and for signals which may follow complex patterns without going under the peak detection threshold.

Detecting presence of a defect. The system offers a choice of two methods to detect presence of a defect or another interesting event in the signal: a sequential detection and a deepest-first detection.

Sequential detection

In this operating mode the signal is scanned sequentially and a presence of a defect is detected when the signal goes out of the threshold area. One possible threshold area is a simple amplitude circle defined by the detect threshold. Another threshold area is used for the so-called directional detection and is shown in Figure 6.14. The simple amplitude threshold works well on a clean signal.

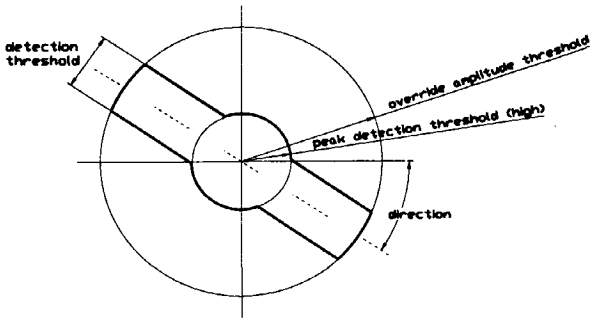


Figure 6.14: Directional threshold and the parameters defining it.

The directional threshold area is useful if oscillation of the signal along some direction has to be ignored (e.g., probe wobble, shallow corrosion).

Deepest-first detection

In corroded pipes it is usually sufficient to detect only several deepest defects. This can be accomplished starting the detection with deepest defects first. In practice, this is done by beginning with the peaks whose phase corresponds to the maximum wall loss. Once such a peak is found the extent of the defect and its real phase are determined. For differential signal, there is a small discrepancy between the phase of a single peak and the phase of the whole defect (between two peaks), therefore, to be sure that the biggest defect has been detected it is necessary to detect several defects using this method. (It is also possible to improve this algorithm to look at two successive signal lobes instead of just one.)

Because inspectors can be also interested in signals of very high amplitude (and with a phase not necessarily corresponding to a big defect), it is possible, for instance after detecting several big defects based on their phase, to switch to detection where signal peaks with the highest amplitude signalise presence of defects.

In order not to waste time on detecting every single defect (which is undesired in case of extensive corrosion), when the system is running automatically, two thresholds can be set:

- wall-loss threshold (in %) — based on the calibration curve — defines the phase range which triggers detection,
- override-amplitude threshold — signals with an amplitude above this threshold are detected even if their phase corresponds to a wall loss below the wall-loss threshold.

Determining extent of a defect. The system has three methods for the determination of the extent of a defect:

- *Connection threshold* — All signal lobes are combined whose edges are within certain distance (the connection threshold) from one another. This is a good method for clean pipes with relatively few defects.
- *Opposite lobes* — Two signal lobes are combined whose peaks form approximately 180 degree angle relative to the origin. The lobes may not be further apart from each other than the connection threshold. This method is useful in pipes with a lot of pitting. The method may not work correctly if complex baffle and defect combination signals are present. (In the prototype it is used only in combination with the deepest-first detection.)

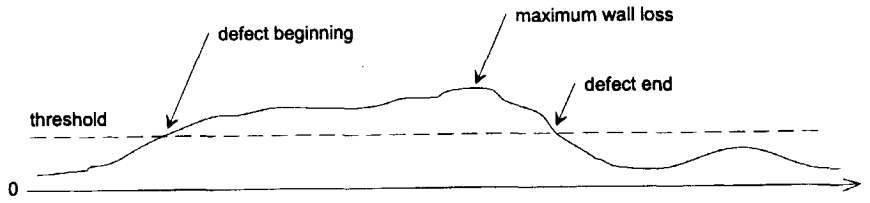


Figure 6.15: The idea of detection of long defects using amplitude of the absolute signal.

- *Case-based extent determination* — This method begins with a single detected signal lobe and then successively extends the detected area with the neighbouring lobes, each time matching the resulting signal against the cases in the case-base. The signal extent which corresponds to the best match found will be returned as the result.

Combined absolute-differential detection. Usually, detection is done in either differential or absolute signal. However, there are situations, for example combination of long defects with pitting, where, theoretically, it could be useful to look at both signals. The system can do it by looking at the absolute signal first, classify it, and then look at the differential signal within the range detected on the absolute signal. Combination of the two classifications lets one draw better conclusions about the type of defect.

This method was tested on some calibration pipes, but no data from real inspection was encountered where it would be useful. A method more useful for handling long wall-loss defects is described next.

Detecting long, uniform damage in absolute signal. It has been observed that long, uniform damage is in practice usually determined using the amplitude of the absolute signal (phase can be used to confirm the inside/outside location of the damage). The idea of the proposed detection method is illustrated in Figure 6.15. For every area where the absolute-signal amplitude is above some chosen threshold a beginning and an end of the area are recorded plus a position and a value of the maximum amplitude within that area and the corresponding wall loss. This method has not yet been implemented in LISSA, but it is relatively simple and, nevertheless, should give good results for cases like erosion and fretting wear.

6.4.4 Frequency mixing

Mixing is done to enable defect interpretation in the presence of a distorting signal from, for example, a baffle or a dent, see Section 2.2.1. In LISSA, mixing is not done for the whole signal (as is usually the case in EC inspection equipment), but only for the possible detected defects whenever no classification can be determined without mixing. Therefore, there is also no mix channel shown on the display all the time, but it is visible only when a possible defect signal has been mixed.

The mixing parameters (the rotation angle α, x and y scale, and x and y offsets) are calculated from two different frequency signals coming from a baffle without a defect. In LISSA they are calculated in two stages:

- first, the angle α and the uniform scale β needed to align the curves are calculated analytically using the method described in [Persoon & Fu, 1977],
- then, all five transformation parameters are calculated using numerical minimisation, the distance being minimised is the Euclidean distance between the curves.

The first stage is necessary to find good starting points for the numerical minimisation; otherwise, incorrect results could be obtained.

The actual mixing is done by subtracting one frequency signal from the transformed signal of the other frequency. The mixing result could be improved if the residual from the mixing-parameter determination was subtracted from the initial mixing result. However, this is not trivial (because of the necessity to properly align the signals) and would significantly increase time necessary to do the mixing.

6.4.5 Phase calculation

The definition of the phase of a defect signal according to the ASME code [ASME, 1995] is an angle of a line through the peak(s) of the signal. In case of absolute data, calculation of the phase is trivial and corresponds to the argument of the complex data point having the maximum amplitude. In case of differential data, phase calculation would also be trivial if all the signals had the same figure "8" shape. However, for more complex signals the determination of the phase can become difficult. At present, the following algorithm for differential signal phase calculation is implemented:

1. a maximum amplitude of the defect signal is found,
2. a threshold is set to 0.25 of the maximum,
3. the peaks of the signal above the threshold are found,
4. distances between all the pairs of the so found peaks are calculated and the most distant pair defines the line whose angle with the real axis is the phase angle.

This algorithm is, however, not entirely reliable. A few complex defects have been encountered for which the phase was calculated incorrectly.

6.4.6 Calibration

A calibration curve is constructed by entering the location (inside ID or outside OD) and the wall-loss (in %) description for the defects detected in the signal. LISSA constructs the calibration curve by fitting parabolas (or lines if less than three points are available) through the phase/wall-loss pairs. Two parabolas are fitted, one for the inside and one for the outside defects. Parabolas are fitted using GLLS (general linear least squares) algorithm [Press et al., 1992]. It is possible to construct a calibration curve for any original and mixed channel (see Figure 6.20), so altogether up to six calibration curves can be constructed simultaneously.

6.5 LISSA — CBR part

The idea of the automatic defect interpretation has already been presented in Section 6.3.2 This section describes the details of the data representations and algorithms used.

6.5.1 Case — short description

A single case in LISSA contains the following information (the details are given in Section 6.5.4):

- The data (defect or some other indication).
- Specification of parameters that should be matched while determining similarity.
- Defect type description corresponding to this indication.
- The action that has to be taken in case of a good match. Two actions are defined: *direct classify*, i.e., simply write the description to a report file, and *use the calibration* curve to determine the wall loss.

6.5.2 Case-base organisation and retrieval

In LISSA the case-base is relatively simple. The main division of the cases is in ones where matching should be done on the original signal and ones matched on the mixed signal. Next, the cases are divided based on the complexity of their signals. The complexity is defined as $^1/_2$ of the ratio of the length of a curve to the diagonal of the enclosing rectangle. It is roughly proportional to the number

of times an EC signal was drawn, i.e., a single defect will have complexity of 1, a double defect will have complexity of 2. Cases of the same complexity are stored in an array.

While searching, the elements of appropriate array are matched sequentially against the current indication. Because determination of the curve complexity is only approximate three neighbouring arrays are always searched.

Given some signal, first a matching case is searched in the "original" cases, if no good match can be found then the search is continued within the "mixed" cases. This is schematically illustrated in Figure 6.16.

6.5.3 Case matching

As shown in Figure 6.16 indications can be matched either on their original signal channels or on the mixed signal channels. While matching any two signal indications, the following three parameters are considered:

- shape of the signal — the particular signal channels to be matched are chosen globally for the case-base,
- phase of the signal,
- amplitude of the signal.

If matching is done on the original signals, two other parameters can optionally be taken into account:

- maximum allowed wall-loss mismatch — only if there is more than one calibration curve for the original channels,
- maximum allowed amplitude of the mixed signal.

Individual parameter matches result in values in the range [0...1]. These values are combined using *minimum* as a fuzzy AND operator.

Matching shape. Many possible ways of Lissajous curve representation and matching have been considered (Section 5.3.3) [van't Hoff, 1996; van't Hoff et al., 1997]. The method finally chosen is presented in the following.

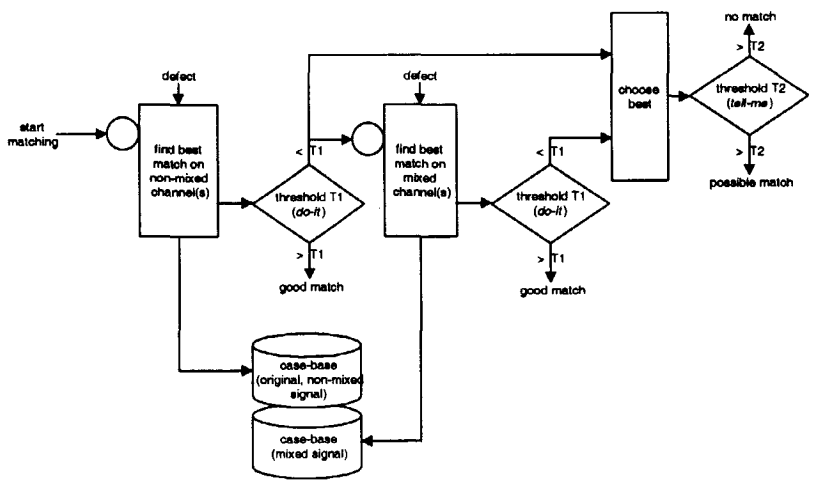


Figure 6.16: Matching scheme (matching of the mixed signal is done only if no good match for the original non-mixed signal can be found).

Originally, Lissajous curves are represented by a series of complex numbers. Before a curve is further processed, it is resampled, that is a curve is re-represented by a constant number of equidistant points. Resampling is necessary because of two reasons:

- It cannot be assumed that the data is sampled with a position-trigger. This means that depending on the speed with which the probe was pulled, scans of the same defect can have various distributions of points on the Lissajous curve.
- The algorithms described in the following require the curve to be described by a constant number of equidistant points.

The resampling is done by first calculating Fourier coefficients for the curve and then reconstructing the curve from these coefficients. Use of Fourier coefficients gives good results as described in [Stepinski, 1990]. The coefficients are calculated using the formula for a polygon described in [Udpa & Lord, 1984]. The number of coefficients necessary to describe a curve is estimated using the formula from [Dekking & van Otterloo, 1986] (bound L with Lebesque constant). The formula usually overestimates the number of necessary coefficients, therefore, an iterative procedure is later used to determine the optimal (as low as possible) number of Fourier coefficients. A maximum allowed distance between points of the curve reconstructed from this optimal number of Fourier coefficients and points of the curve reconstructed from the L-bound number of Fourier coefficients may not be higher than 1% of its diagonal or a value roughly equal to the amplitude of noise (whichever is smaller).

A curve is represented by all its resampled points. In order to be able to compare curves of various sizes, positions, and orientations, the curves have to be transformed first, this is done as follows:

- The position of a curve is normalised by setting the zeroeth Fourier coefficient to zero before resampling.
- The start point is normalised by choosing resampled point lying between the ends of the curve and being closest to the origin. The Fourier coefficients have to be adjusted for this change.
- The difference of the scale and the rotation of two curves described by vectors A and B is compensated by multiplying Fourier coefficients of one of the curves (say B) by a complex number $\beta e^{i\alpha}$ which is obtained by minimisation of the distance between A and $\beta e^{i\alpha}B$. The minimisation is done analytically using the method described in [Persoon & Fu, 1977].

After transformation, the two curves are compared by calculating the maximum distance between the vectors of the resampled points. The value of the maximum distance is expressed as a fraction of the diagonal of the curve-enclosing rectangle. This value lies in the 0...1 range, it is subtracted from 1 to give the final value of the shape match.

Phase-angle matching. The shape of the Lissajous curve alone is not sufficient to distinguish between various defects. For example, some baffle or baffle-with-defect signals can have the same shape as a signal from a single defect. However, they can be distinguished if also the phase of the signals is taken into consideration — in one or more frequency channels if necessary. When entering parameters of a new case, it is, therefore, necessary to specify the angle range in which other cases should fall in order to match with the given case. There are three ways the angles can be matched:

- the angle of the signal does not matter (i.e. all angles match),
- the angle should lie within some range — user can specify it by giving absolute range values or by giving the range relative ($\pm$) to the angle of the given signal,
- the angle should fall within the calibration curve range for the given channel — this option will generally be used for defect signals.

The simplest way to check the angle match would be to return 1 (true) whenever the angle falls within the allowed range and 0 (false) otherwise. This approach is sufficient if one is interested only

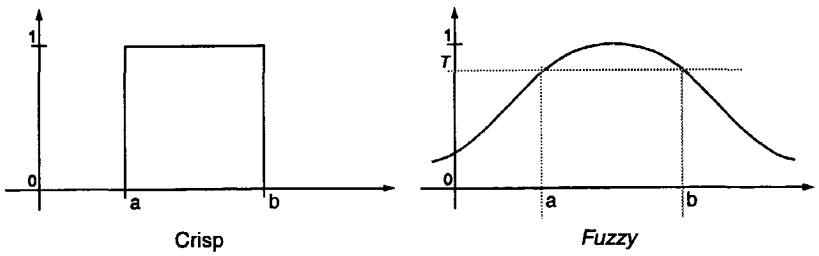


Figure 6.17: Crisp vs. fuzzy matching.

in good matches. However, the system can also use slightly worse matches for suggestions to the user. For this purpose it is advantageous to know how close an angle is to the allowed range. This problem is solved using a fuzzy membership function to represent angle match result. This is illustrated in Figure 6.17. When the angles are within the allowed range, the match is higher than the acceptance threshold *T1*, when the angles are outside of that range, the match value gradually decreases. This fuzzy-matching method is also used for the calibration angles.

Checking the wall-loss match. Some non-defect signals may have the same shape as defect signals and their phase angles may lie within the calibration range. These signals can still be distinguished from defect signals if the scan has been done in two frequencies and for both frequencies calibration curves have been made. The difference lies in the fact that defect signals will give (almost) the same defect location (ID/OD) and wall-loss results for both frequencies; the wall-loss results for non-defect signals will not match.

The amount of the allowed wall-loss mismatch is specified with each case. It is matched using the fuzzy method as described for the angle ranges.

Amplitude matching.

All signals

Generally, if very small signals (thus distorted by noise) are matched with very big signals may lead to undesired results. Therefore, the system will match only shapes whose amplitude ratio lies between 0.5 and 2.0. Also here, the limits are not hard but fuzzy.

Checking maximum allowed amplitude of the mix signal

Sometimes, the shape of the original signal may not be sufficient to distinguish reliably between baffles and baffles with very small defects. The difference is, however, visible in the amplitude of the mix signal. Therefore, for signals like baffles it is possible to specify the maximum allowed amplitude of the mix signal (the limit is also fuzzy).

6.5.4 Case — detailed description

Previous section has presented various ways case parameters can be matched. The actually used matching procedure is determined by the case parameters. The full content of a case is as follows:

- The original (and mixed, if mixing was done) data of all the channels within the detected limits. (Storing the original data is necessary for the verification.)
- The resampled points of the curve for the channel(s) that will be matched.
- The Fourier coefficients of the curve — used to optimise for rotation and scaling.
- The definition of the angle range that has to be matched. It can be:
  - all angles,
  - a to b angle range,
  - ± range,
  - use the angle within the calibration (if a two-frequency signal is stored in the case then the allowed wall-loss mismatch has also to be entered).

- Maximum allowed mix signal amplitude (if needed).
- Defect type description — consists of a main description and a sub-description, for instance, "defect (small)." It will appear on the reports and can be used to estimate good *T1* threshold (see next section) and to check for case-base consistency.
- The action that has to be taken in case of a good match. Two actions are possible:
  - direct classify — simply write the description to the report,
  - use a calibration curve to determine the wall loss.

6.5.5 Threshold estimation

An approximate value of the acceptance threshold *T1* can be estimated after the case-base has been filled with representative cases (usually after the calibration). For this purpose two histograms of the values of matches between cases in the case-base are constructed: one for all the cases of the same type and another for all the cases of different type. This is done twice: first, the type is defined by only the main case description, and second, the type is defined by a combination of the main and sub-descriptions. The resulting histograms are shown in Figure 6.18. Over the histograms two lines are drawn corresponding to thresholds *T1* ("do-it") and *T2* ("tell-me"). Such histograms can be used to choose the correct threshold values. This method will work if there are no inconsistencies in the case base.

To better analyse the data, it is possible to choose what matches should be displayed on the histograms. The possibilities are:

- matches of curve shapes,
- matches of the phase angles and amplitudes,
- all of the above (as actually used by the system).

It is also possible to define how many matches of each of the defects should be put into the histogram — the *K* factor. If *K* is 1 then for each defect only one best match will be stored in each of the histograms, for *K* equal 2, two best matches will be stored, etc. Doing the histograms for several values of *K* gives an idea about the extent of the signal classes.

6.5.6 User interface

In the prototype, the data can be displayed as a line plot and as a polar plot, see Figure 6.19. There is also a zoom plot that shows magnification of the signal under the cursor. It is possible to resize the windows in which the data is displayed. Different channels are displayed in different colours. The detected end-distortions (see Section 6.4.2) are displayed in a colour different from the rest of the signal. It is possible to select which signals should be displayed. The scale for the differential and absolute signals can be set by the user or it can be set automatically to the maximum value between the signal edges.

As already mentioned, this type of interface is not appropriate for real inspections. The experience with the prototype suggests that one should limit the usage of a mouse. Also, resizable windows are not useful for a field version, a choice of several predefined layouts would be better. One should also reserve part of the screen for fixed parameter-entry window as pop-up dialog boxes obscure the signal display.

6.6 Tests

6.6.1 Test data and experiments

The data used for the system evaluation comes from real measurements (not from simulation) and can be divided into three groups as far as their origin is concerned:

- data from calibration pipes — 4 data sets,
- data from pipes pulled out from a heat exchanger — 1 data set,
- data from real field heat-exchanger inspections — 6 data sets.

Some of these data sets were too small and some not sufficiently documented. In this section only tests on data sets that provide meaningful results are described. These are the following four:

- *ASME calibration pipe (cal* files)* — An ASME calibration pipe was scanned once clean and then several times with a ring simulating a baffle over various spots of the pipe. In total 11 scans were made. All measurements were made with two frequencies (200 kHz and 100 kHz), the settings were constant except for the gain.
- *Test heat exchanger (tk* files)* — This data was obtained from measurements of a small heat exchanger present at the TMT - Kontroll Technik lab. The heat exchanger has extensive baffle damage caused by vibration but has no corrosion damage. Most of the defects present are grooves under baffles. The tubes are made from CuNi, have outside diameter 25.2 mm, inside diameter 21.0 mm and wall thickness 2.1 mm. There are 18 baffle plates per tube, some of which are half baffle plates. The scan was done with two frequencies 60 and 30 kHz. The signal has very low levels of noise.

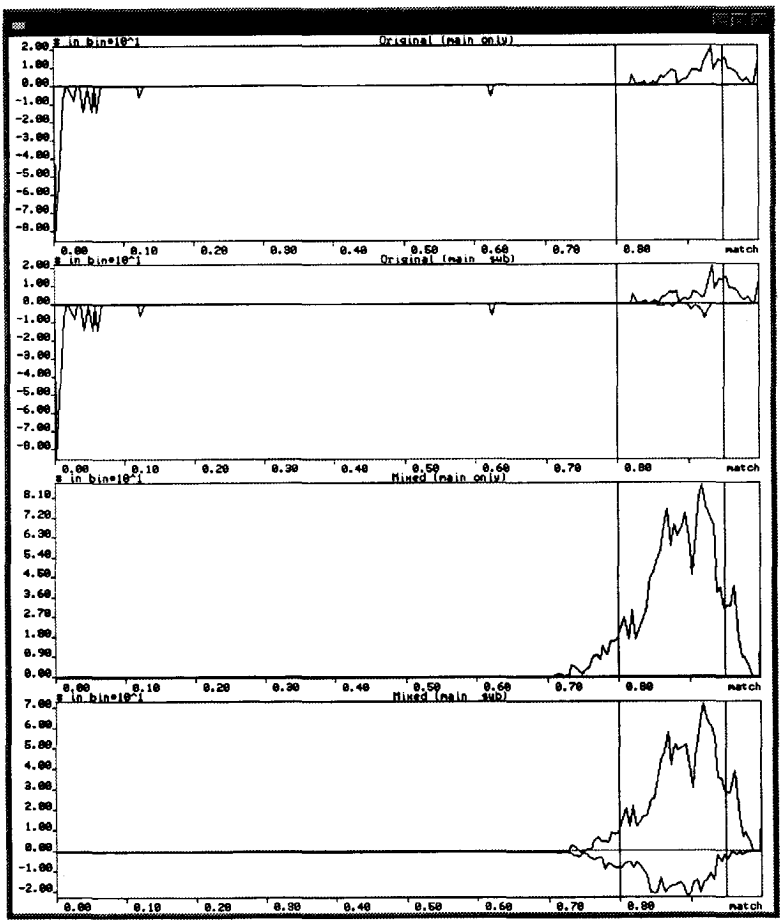


Figure 6.18: Case-base match histograms. Eight histograms are visible; for original (top two) and mixed (bottom two) signal cases, for main-only (first and third) and main-and-sub (second and fourth) descriptions, and for same (positive values) and different (negative values) classes.

- *Welded pipes (va* files)* — This data comes from a heat exchanger with thin Austenic (1.4439) pipes (rolled and welded). The outside pipe diameter is 28 mm and the wall thickness is 0.7 mm. The frequencies used during the scan are 400 and 800 kHz. The data contains a continuous, zigzag-like signal from the weld, apart from that there are baffles, dents, some defects, and ferritic inclusions.
- *Condenser tubes (b* files)* — The data comes from measurements made on a condenser of the nuclear power plant Philippsburg 1. The pipes are made of brass 76. They are 13.80 m long, the outside diameter is 28 mm, the inside diameter 26 mm, and the wall thickness 1 mm. There is a thinning from inside present, this is a result of cleaning after corrosion (the thinning is bigger at one end of the pipe). The thinning influences signal from the baffles. The amount of thinning is being determined using the absolute signal. During the original measurements three frequencies were used: 50 kHz for all defects, 100 kHz used mainly for internal defects, and 25 kHz used mainly for external defects. Our data had only 50 and 100 kHz channels. Two calibration scans were made, one for the differential signal, and one on the amplitude of the absolute signal.

The experiments done with the data generally simulate actions that would take place during normal inspections. Thus, first correct system settings are chosen on basis of calibration data, then calibration curves are constructed, and, finally, the data files are scanned resulting in a report of defects found. In the following sections the results of these experiments are described, arranged per data set as described above.

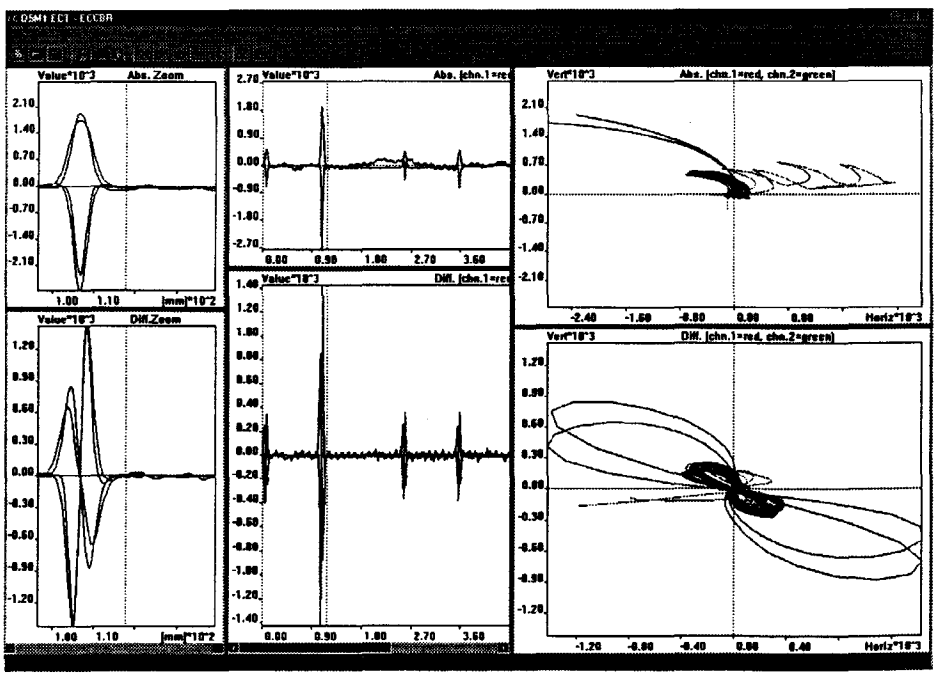


Figure 6.19: Interface of the LISSA prototype. From left: the zoom plots, the line plots, and the polar plots.

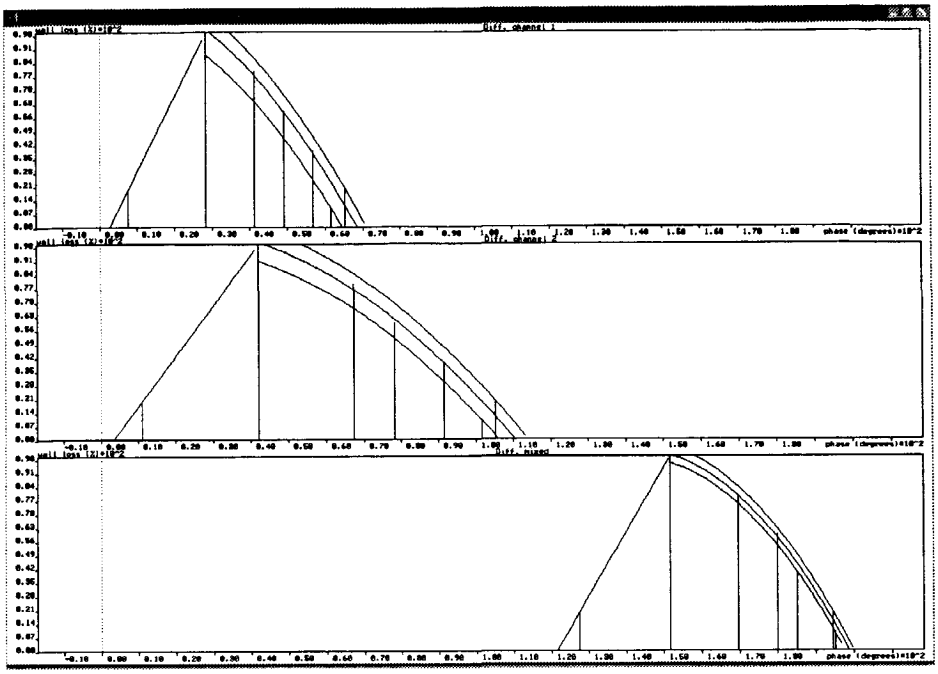


Figure 6.20: Calibration curves constructed from call file, two original signal channels and the mix channel.

6.6.2 cal* files

The cal* files contain scans of one and the same ASME calibration pipe (see Figure 6.1). The difference between the scans is the location of a ring simulating a baffle. The data comes from an absolute probe (2 frequency channels) and the differential signal has been calculated from the absolute one. Because of the localised character of defects (pitting) only the differential signal is used for this experiment. An example of the signal can be seen in Figure 6.7.

Two data files were used for calibration. One contained the following defects: ID20%, OD10%, OD20%, OD40%, OD60%, OD80%, 1.3 mm hole, 0.80 mm hole; and the other contained a baffle signal. The system was put in the calibration mode and, first, the mixing parameters were determined using a signal from the baffle over a non-defect spot. The parameters found were: $T_x=6$, $T_y=8$, $\theta=53°$, $S_x=2.39$, $S_y=1.71$. Next, one data file was scanned and every detected defect (except the smaller hole, because of a less accurate phase determination on it) was used to construct the calibration curve (it was constructed for #1, #2 and mix differential channels), see Figure 6.20.

Next, the match threshold *T1* was set to 0.99, calibration mode was switched off, and the calibration file was scanned again. This time all the nine defects were entered into the case-base (where applicable a wall-loss mismatch of 15% was allowed). For the first defect, for the small hole, and for the baffle full case description had to be entered, for the remaining indications system-suggested descriptions were used. For the obtained case-base, a histogram of matches was made and based on it the *T1* threshold was set to 0.925 and *T2* threshold was set to 0.75. The results of this and further scans are presented in Table 0.1. As can be seen the bigger defects were mostly recognised automatically. The small defect match was mostly below the *T1* threshold and thus recognised as

Table 0.1: Results of the scan of cal* data. The drawing of the calibration pipe is in Figure 6.1. Example signal is shown in Figure 6.7. Abbreviations: N/R — not recognised, s — similar, cb — entered into case-base.

File	defect a	defect b	defect c	defect d	defect e	defect f	defect g	defect h	extra
real defect wall loss	ID20% (groove)	OD10% (groove)	OD20% (4 holes)	OD40%	OD60%	OD80%	100% (big diameter)	100% (small diameter)	
cal1	N/R cb: defect ID20%	s: defect OK OD23%	s: defect OK OD12%	s: defect OK OD39%	s: defect OK OD62%	s: defect OK OD77%	s: defect OK 100%	s: defect cb: defect small 100%	
cal1b1	auto: defect ID18%	auto: defect OD18%	auto: defect OD7%	auto: defect OD34%	auto: defect OD47%	auto: defect OD62%	auto: defect OD94%	s: defect small OK OD88%	N/R cb: baffle
cal2b1	auto: defect ID19%	auto: defect OD27%	auto: defect OD10%	auto: defect OD42%	auto: defect OD67%	auto: defect OD78%	auto: defect 100%	s: baffle cb: small defect under baffle OD77%	
cal3b1	auto: defect ID16%	auto: defect OD29%	auto: defect OD13%	auto: defect OD40%	auto: defect OD63%	auto: defect OD77%	N/R cb: defect under baffle 100%	auto: defect small 100%	
cal4b1	auto: defect ID18%	auto: defect OD32%	auto: defect OD12%	auto: defect OD40%	auto: defect OD60%	s: defect under baffle OK OD73%	auto: defect 100%	s: defect cb: defect small OD96%	
cal5b1	s: defect OK ID18%	auto: defect OD29%	auto: defect OD11%	auto: defect OD41%	N/R cb: defect under baffle OD61%	auto: defect OD78%	auto: defect 100%	s: defect small OK OD96%	
cal6b1	s: defect OK ID19%	auto: defect OD29%	auto: defect OD16%	N/R cb: defect under baffle OD41%	auto: defect OD65%	auto: defect OD81%	auto: defect 100%	s: defect cb: defect small OD98%	
cal7b1	auto: defect ID17%	auto: defect OD28%	N/R cb: defect under baffle OD10%	auto: defect OD43%	auto: defect OD65%	auto: defect OD77%	auto: defect 100%	s: defect small OK ID99%	
cal8b1	s: defect[a] OK ID19%	s: defect under baffle OK OD9%	auto: defect OD12%	auto: defect OD43%	auto: defect OD65%	auto: defect OD79%	auto: defect 100%	s: defect cb: defect small 100%	
cal9b1	s: defect under baffle OK ID12%	auto: defect OD29%	auto: defect OD13%	auto: defect OD41%	auto: defect OD65%	auto: defect OD80%	auto: defect 100%	s: defect small OK 100%	
cal10b1	auto: defect ID16%	N/R cb: defect under baffle OD5%	auto: defect OD12%	auto: defect OD44%	auto: defect OD68%	auto: defect OD77%	auto: defect 100%	s: defect small OK 100%	

a. possibly outside calibration range

only similar. The baffle-plus-defect indications were either recognised as similar or not recognised at all. No mistakes were made.

This test was quite easy for the system because the differences between the defect indications were small — all the eleven scans were done on the same tube. Still, it showed that the system performed as expected and that now experiments with data from real inspections could be done.

6.6.3 tk* files

The tk* files contain two-frequency differential signals. The signal-to-noise ratio is high. Most of the scanned pipes have 18 baffles under most of which there are defects, therefore, mixing has to be done. An example signal can be seen in Figure 6.21.

The system was set in the calibration mode and the mixing parameters were determined. The parameters found were $T_x=-2$, $T_y=-1$, $\theta=56°$, $S_x=2.33$, $S_y=2.19$. Next, the signal from the calibration pipe was scanned and every detected defect (except one defect which had a different form and a dent) was used to construct the calibration curve (the calibration curve was constructed for #1, #2, and mix differential channels similar as in case of cal* files).

Then, the match threshold *T1* was set to 0.99 and the calibration file was scanned. All the ten defects and one dent indication were entered into the case-base. For the case-base a histogram of matches was made and based on it the thresholds *T1* and *T2* were set to 0.93 and 0.75 respectively. Next, the remaining files were scanned. The summary of results is in Table 6.4. The meaning of the labels used for the various categories is explained in Table 6.3.

As can be seen, on average 82% of all detected flaws were classified automatically. As far as we can verify all automatic classifications were correct. Figure 6.22 shows how the percentage automatically classified indications progressed during the scan. The "dip" visible beginning with the file tkx12y11 is caused by the first occurrence of small-baffle signal in the data, the system has to learn to recognise them. Beginning with file tkx13y8 the system has already learned how to recognise small baffles.

If the threshold *T1* is lowered then the system could classify more indications automatically even up to 97% but then the "similar → CB different" indications would end up incorrectly classified. If one checks the results then it turns out that the best match in the "similar → CB different" column was 0.908 and it was a dent recognised as similar to a defect under a small baffle. If one checks the results for all "similar" flaws then it turns out that 20 of them had a match above 0.908. this means that an optimal (a-posteriori) threshold, for this data set, would give us an automatic classification ratio of about 90%.

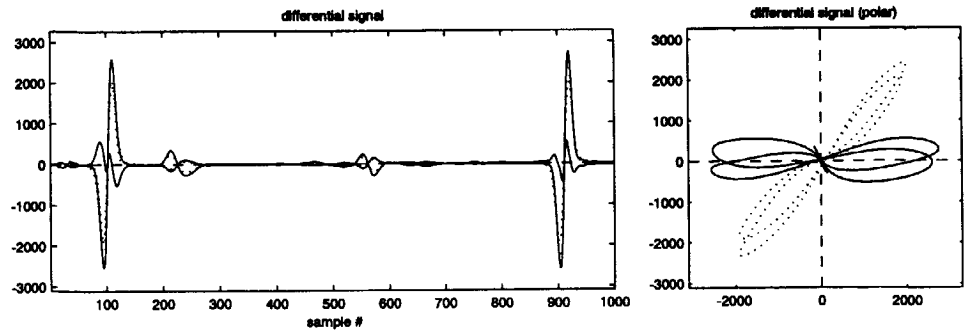


Figure 6.21: Example of tk* data — two baffles with baffle damage (at 100 & 900) and two dents (at 200 & 600). The dotted signal is from the second frequency.

Table 6.3: Meanings of the column labels used in the tables with scan-result summaries (Table 6.4, Table 6.5, and Table 6.6).

label	meaning
automatic	automatically recognised
similar → OK	correctly recognised as similar to something, operator only had to click OK
similar → CB same but adjusted	recognised as belonging to the correct general class but either defect extent or precise case description had to be adjusted
similar → CB different	recognised as similar to a wrong class, case description and, if necessary, also extent had to be changed
similar → don't add to CB	incorrectly recognised as similar to something, was given a probable description but not stored in the case-base because unsure what it is, or too complex, or could corrupt case-base, etc.
similar → skip	most probably a spurious detection
not recognised → CB	new case entered into case base
not recognised → don't add to CB	not recognised (further see similar → don't add to CB)
not recognised → skip	most probably a spurious detection

Table 6.4: Results obtained on tk* files.

file	flaw count	automatic	similar -> OK	similar -> CB same but adjusted	similar -> CB different	similar -> don't add to CB	similar -> skip	not recognized ->CB	not recognized -> don't add to CB	not recognized -> skip	check sum	automatic + similar->OK	entered into case-base
tkx10y5	10	8	1	0	0	0	0	1	0	0	10	9	2
tkx11y8	20	6	2	1	0	0	0	1	0	0	10	8	4
	30	9	1	0	0	0	0	0	0	0	10	10	1
tkx12y11	40	10	0	0	0	0	0	0	0	0	10	10	0
	50	6	1	2	1	0	0	0	0	0	10	7	4
tkx12y5	60	5	1	0	1	0	0	2	0	1	10	6	4
	70	5	4	0	0	0	0	0	0	1	10	9	4
tkx13y4	80	4	4	0	1	0	0	1	0	0	10	8	6
tkx13y8	90	9	0	1	0	0	0	0	0	0	10	9	1
tkx1y8	100	7	2	0	1	0	0	0	0	0	10	9	3
tkx2y5	110	9	1	0	0	0	0	0	0	0	10	10	1
	120	9	1	0	0	0	0	0	0	0	10	10	1
tkx2y7	130	9	1	0	0	0	0	0	0	0	10	10	1
	140	10	0	0	0	0	0	0	0	0	10	10	0
tkx3y8	150	8	2	0	0	0	0	0	0	0	10	10	2
	160	9	1	0	0	0	0	0	0	0	10	10	1
tkx4y5	170	10	0	0	0	0	0	0	0	0	10	10	0
	180	6	3	0	0	0	0	0	1	0	10	9	3
tkx4y7	190	10	0	0	0	0	0	0	0	0	10	10	0
tkx5y8	200	10	0	0	0	0	0	0	0	0	10	10	0
	210	9	1	0	0	0	0	0	0	0	10	10	1
tkx9y8	220	10	0	0	0	0	0	0	0	0	10	10	0
	230	10	0	0	0	0	0	0	0	0	10	10	0
sum		188	26	4	4	0	0	5	1	2	230	214	39
% of total		82%	11%	2%	2%	0%	0%	2%	0%	1%	100%	93%	17%

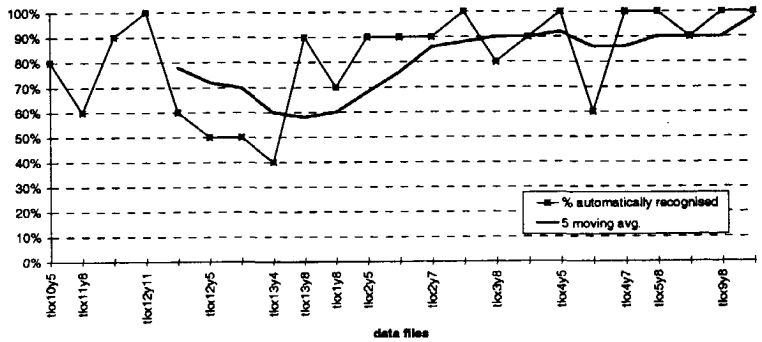


Figure 6.22: Progress of the amount of automatically classified indications while scanning tk* files.

6.6.4 va* files

The va* files contain two-frequency differential signals. The signal-to-noise ratio is relatively high. The baffle signal is clear so that mixing can be reliable. There is some small noise-like signal from the weld in the pipe but it can be ignored if directional detection is used. An example va* signal is shown in Figure 6.23.

The mixing parameters calculated from the baffle signal in the calibration file are: $T_x=-2$, $T_y=2$, $\theta=64°$, $S_x=2.05$, $S_y=2.99$. The initial case-base was also built from the data in that file. For the scan the *T1* threshold was set to 0.95. Summary of test scans is shown in Table 6.5. Because the threshold was set higher than in tk*, we see that the percentage of the automatically classified is smaller then in case of tk* data while the percentage in the "similar → OK" class is higher.

In Figure 6.24, showing how the percentage of automatically classified indications progressed during the scan, three major dips are visible. One corresponds to the vax5y25 file which has weld indications at a different phase angle than in the other files. This results in many spurious detections. The second dip corresponds to the vax9y101 file which has a scan of pipe with clamps in addition to baffles, the system does not recognise them when it sees them for the first time. The last dip corresponds to the vax15y13 file which has data from ferritic inclusions. They are not recognised at first; moreover, it was unclear to us what a ferritic inclusion indication should look like, that is why not all detected indications were entered into the case-base and the system did not learn them.

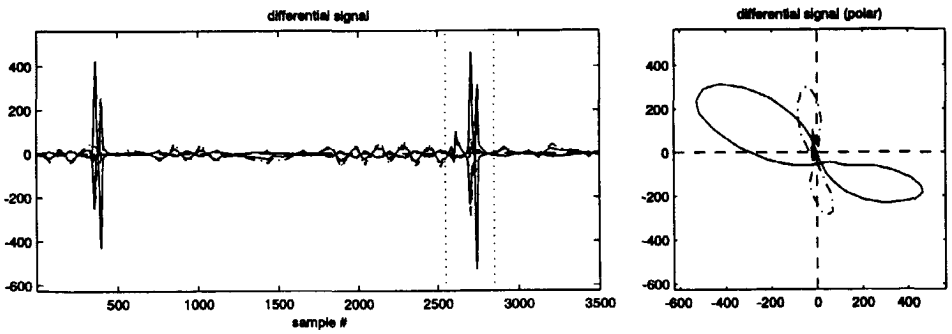


Figure 6.23: Example of va* data. The big signal is from a baffle, the "noise" is caused by a weld in the tube. Near the second baffle (at sample #2600) a small, but deep, defect is visible.

Table 6.5: Summary of the scan results for va* data.

	automatic	similar -> OK	similar -> CB same but adjusted	similar -> CB different	similar -> don't add to CB	similar -> skip	not recognized ->CB	not recognized -> don't add to CI	not recognized -> skip	total	automatic + similar->OK	added to case-base
sum	117	104	5	2	1	7	34	26	54	350	221	145
% of total	33%	30%	1%	1%	0%	2%	10%	7%	15%	100%	63%	41%

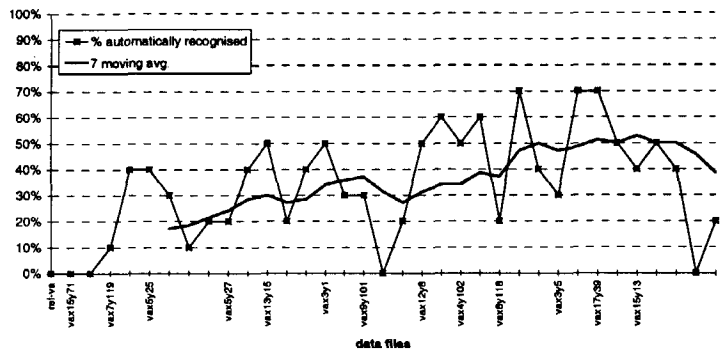


Figure 6.24: Progress of the amount automatically classified indications while scanning va* files.

6.6.5 b* files

The b* data comes from condenser pipes with pitting. Because the pipe diameter changes slowly, one absolute channel was used to monitor that change (in this experiment this absolute channel is not used). The two-frequency differential signal contains indications from baffles, but no mixing is done. This is because the shape of baffle signal changes as the wall thickness varies, and because the shapes of baffle signals in the two channels differ too much from each other (in one channel the baffle indications are very flat). An example of b* data is shown in Figure 6.25. Because there is a lot of pitting, case-based deepest-first detection was done.

The calibration file contains data from several pipes with various defects, it was used to construct the calibration curve. The 20%OD and 20%ID defects were not detected with the detection amplitude set to detect the small hole defect as it was done for the real inspection. After calibration, the file was scanned to the case-base. Next, all the B* files were scanned with the deepest-first detection method. Summary of b* file scans is shown in Table 6.6. The best match of the "similar → skipped" was 0.914 (for a mistaken baffle). The best match for "similar → CB different" was 0.938 ("defect irregular" instead of "defect with baffle"). The best match for "similar → don't add to CB" was 0.949 (it was a "distorted defect at pipe exit" — similar to "defect irregular"). The progress of the average percentage of automatically classified indications is shown in Figure 6.26.

One can clearly see the percentage rising from approximately 20% to approximately 50%. The first dip is caused by unrecognised baffles in files b3x6y36 and b1x8y6. The second dip is caused by baffles at the beginning of files b1x2y36 and b1x2y13 — these baffles were, however, correctly recognised as similar to other baffles.

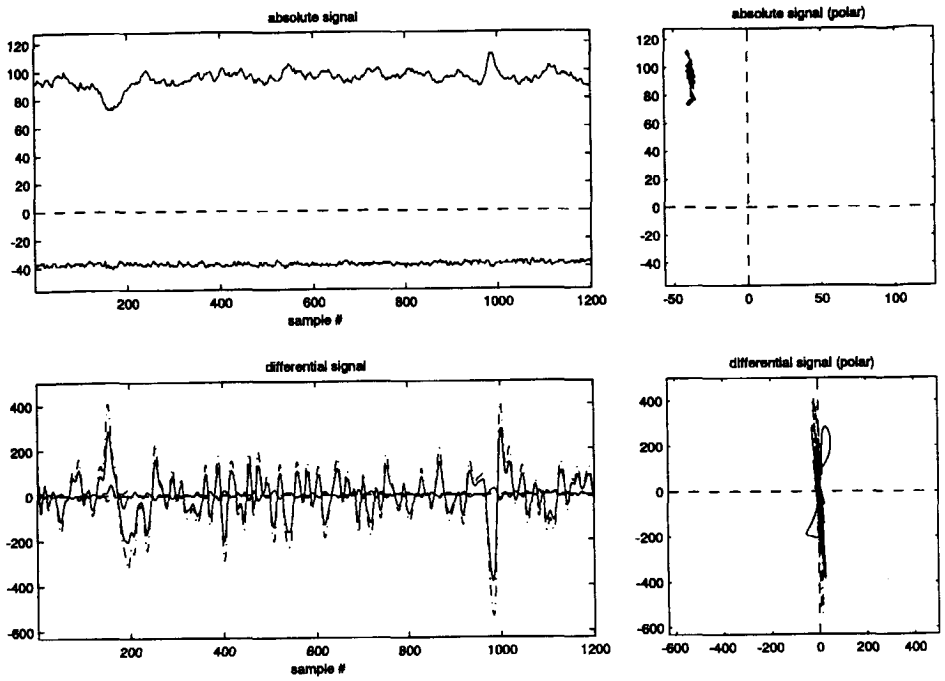


Figure 6.25: Example of b* data.

Table 6.6: Summary of the deepest-first scan results of b* data.

	automatic	similar -> OK	similar -> CB same but adjusted	similar -> CB different	similar -> don't add to CB	similar -> skip	not recognized ->CB	not recognized -> don't add to CI	not recognized -> skip	check sum	automatic + similar->OK	added to case-base
sum	397	380	125	16	21	23	128	61	19	1170	777	649
% of total	34%	32%	11%	1%	2%	2%	11%	5%	2%	100%	66%	55%

The correctness of automatic classifications was checked for the last two files in the series. At the end of the scan the case-base contains the most cases, therefore, there is the highest chance of making a mistake by matching with a wrong case. In the last file b10x2y40, 27 defects were classified automatically, no mistakes were made though in three cases a part of a multiple defect was classified as a single defect. In the file b10x4y20, 31 defects were classified automatically, there were also no mistakes though, again, in two cases defects classified as single should have better been combined and classified as multiple defects.

6.6.6 Discussion of the test results

The data. First of all it should be noted that the experiments described above are certainly not realistic as far as the amount of data is concerned. In the experiments at most 25 pipes from the same heat exchanger were scanned while in reality even a small heat exchanger would have at least twice

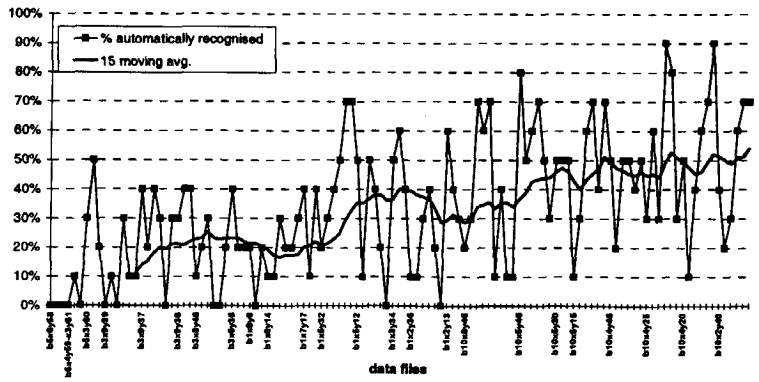


Figure 6.26: Progress of the amount automatically classified indications while scanning B* files using deepest-first detection method.

as many, and usually the number of pipes would be in the order of hundreds or thousands. This point is important as the performance of LISSA improves as it learns from the pipes already scanned. On the other hand, more data would imply a greater case-base size that could lead to long search times and possible inconsistency problems.

From the data sets that were available tk* and va* data sets seem to give a good idea how the overall performance of the system would be in a real inspection situation. This is because the tubes were more or less randomly chosen and represent well the kind of data that would be available during real inspection. va* data is more complex than tk* data and it seems that the available data sample was not big enough for the system to reach a high recognition ratio. It is possible that b* data is also representative for real inspection; however, at least $^1/_3$ of the data was chosen with extraordinary big indications, and another $^1/_3$ of the data has very small indications.

Other data sets were also used for testing (reported in [Jarmulak, 1997]) but they were not representative for real inspection: either the data sets were small, or they were chosen with some specific defects in mind, or baffle indications were missing.

Results for not corroded pipes. From the results of experiments on tk* and va* files one can see that the system operates well on non-corroded pipes. In such pipes defects are clearly separated from one another and the shapes of indications are not too much distorted by noise. The extent of defects can be clearly determined and their form is highly repeatable. In such a situation LISSA can do fully automatic classification already after having seen a few example files. Especially pipes with few defects (mostly baffles) can be quickly evaluated. In pipes with many defects a detailed analysis of each defect can be done much faster than an expert could do it. Whenever some anomaly is detected then the system alerts the operator (as was the case with ferritic inclusions in va*). Also, weak noise, like that from a weld in va*, is not much of a problem for LISSA because it can be ignored by choosing the detection method appropriately.

Results for pipes with high-density pitting. It turns out that LISSA has problems with processing data from pipes with high-density pitting (heavily corroded). In such case a human operator is faster than our system. The reason for this is that a human operator usually ignores most of the data from such pipes concentrating only on exceptional signals. The system is capable to do deepest-first detection thus theoretically looking at the biggest defects first. However, it is still necessary to look (either automatically or with assistance of the operator — depending on the signal) at several defects per file to be sure that the biggest one has been detected. Because of the way the algorithm is implemented and because of the distribution of the phase of the signal, in many situations the

algorithm will detect baffles before it goes over to the deepest defect. Ideally these baffles should be classified automatically (as has been observed on some b* files); however, if they are distorted then the operator will have to check the classification for these baffles.

Signals in b* files are *flat* (almost lines) making it necessary to use a higher *T1* threshold to distinguish between various indications and resulting in more work for the operator. Some suggested solutions for this problem can be found in Section 6.7.2.

Speed. As far as the speed is concerned one can distinguish the speed of the algorithms used (e.g., shape matching) and the overall speed with which an operator using the system can scan the data.

As far as the algorithm speed is concerned, for data sets like tk* it is acceptable (a 300 MHz Pentium computer with 64MB RAM was used to carry out the test). It takes only a few seconds to load each file, and about 10 seconds to scan a file fully automatically (that is including writing all the results to the report file). For bigger data files, like b* with more than 40000 data samples, the file loading may take up to 30 seconds because of all the processing (mainly drift removal) that has to be done on the data. The speed of the case-base searching algorithm is acceptable, though at the end of the sequential b* scan, when almost 900 cases had been entered into the case-base, the search times were perceived as too long. It should be noted that during the development no special attention has been paid to speeding up the case retrieval.

As far as the overall speed of inspection is concerned it is influenced by many factors. If one assumes the same number of detections and the same percentage of automatically classified indications then the main factor influencing the inspection speed will be the design of the user interface. In the prototype the user interface is rather rudimentary (see discussion further); as a result, the overall speed of the inspection was not good.

Reasons for non-optimal results. One of the experiments not described here (described in [Jarmulak, 1997]) has shown that, for example, incorrect calibration can lead to much fewer automatically recognised indications. Also, change in the parameters of the signal, for example, a slightly different phase of one of the channels, leads to worse results because the system is forced to learn to recognise new indications. Pipe entry and pipe exit artefacts may lead to spurious detection or to distortion of real defects. Varying parameters of a pipe (e.g., changes in a weld) can also defeat the directional detection algorithm leading to many spurious detections.

Fortunately, in the experiments none of these problems has lead to mix-up in the classes of the recognised flaws. However, these problems do slow down the whole inspection, requiring more work to be done by the operator.

Problems due to the use of a prototype. During experiments several things did not work as well as they could have, mainly because the current software is still in a prototype stage, meaning that not all the functionality is implemented as it should be in the final application.

User interface

The whole process of scanning a file could be made several times faster by improving the user interface, especially the following:

- When the "new case" dialog box pops-up it usually covers the signal, thus usually has to be moved away. A solution to this problem could be to reserve part of the screen for the input area.
- When a user has to determine the extent of the signal from a flaw, he has to use buttons to add neighbouring signal lobes. Sometimes, the buttons have to be clicked several times. Also, in some rare cases one would like to "undetect" part of the signal which is not possible. A more convenient and flexible method of flaw extent determination would have to be implemented, for instance, using two line cursors.

- The scale of the displays cannot be changed during scanning. This is inconvenient when both high and low amplitude indications are present. Ideally, one would want the scale to be changed automatically for each detected flaw.

Phase determination

In a few cases the phase of the signal was not correctly determined by the system. This may reduce the reliability of the system. A more reliable phase determination algorithm would have to be implemented.

One lobe indications

These indications correspond most probably to an edge of a defect. When the case-based flaw extent determination was used we could not enter such indications into the case-base because later they could match against half of full-defect indications. One could either ignore this problem (such indications are not common) or require that they can only match against other indications if they are isolated (in a sufficient distance from any other indications).

6.7 Assessment

In this section the results of the experiments are discussed in the light of the earlier stated requirements (Section 6.1.4 & Section 6.3.1). As will become clear these requirements are interrelated and often optimising the working according to one criterion will compromise the other ones.

6.7.1 Fulfilling the requirements

Reliability. As the system does require user input (not all decisions are taken automatically) the overall reliability of the system will depend on the accuracy of the judgements and decisions made by the operator. In this section the problem is simplified by assuming that the operator is always right.

100% detection

100% detection is possible if one assumes that any significant signal will have a higher amplitude than the noise. In this respect LISSA is not different from the existing commercial EC software. In some situations fulfilling the requirement of the 100% detection may result in a great amount of data to be analysed. This is evident for data from heavily corroded pipes. In some experiments (reported in [Jarmulak, 1997]) where the b* files were scanned sequentially, up to 400 defects (corrosion pits) were detected per pipe. A human operator is incapable of analysing that amount of data within reasonable time. Doing part of the analysis automatically may help.

100% correct

As far as the correctness of classifications is concerned two cases can be distinguished: the classifications made automatically by the system, and the classifications made by the user.

The correctness of the automatic classifications depends mainly on the choice of the threshold *T1*. The higher the threshold the more confidence one can have in the automatic classifications but the number of indications automatically classified will be small. Setting the *T1* threshold low will give more automatic classifications but a chance that they are wrong will be higher. In the experiments the *T1* threshold was given the following values (percentage is for the whole scan, for scans of larger data sets it is higher than this at the end of the scan):

- 0.908 — tk* — a-posteriori threshold, would give an estimated 90% automatically classified,
- 0.930 — tk* — 82% automatic (93% automatic & similar),
- 0.950 — va* — 33% automatic (63% automatic & similar),
- 0.950 — b* deepest-first — 34% automatic (66% automatic & similar).

As far as we can verify all automatic classifications made during the experiments were either correct or the error made was insignificant — usually in the type of defect and not in the depth of the de-

fect, for example, a big defect under small baffle was classified as defect, or a part of a multiple defect was classified as a single defect.

The correctness of classifications made by an operator using LISSA depends to a large degree on the presentation of the data, therefore, it is important that the user gets all the information he needs to classify a defect. Also, when the system makes suggestions it is important to show certainty of the system in the suggestion made — this will usually be the value of the match.

Operator overload reduction. In common inspection practice often the requirement of 100% detection is abandoned in order to keep the amount of work that the operator has to do in manageable limits. This is, for instance, the case with heavily corroded pipes. For such pipes only a few deepest defects are detected. As we could see, LISSA can also do deepest-first detection, though the experiments done were not fully realistic because often per pipe tens of defects were detected (instead of only a few) to get a statistically significant amount of results.

Automatic classification

In an ideal situation the system should be able to classify all the indications automatically (after a short period of learning). An almost ideal behaviour can be seen in the experiments with tk* files. However, with some types of damage this is not possible. Decreasing the *T1* threshold to increase the amount automatically classified may in turn lead to misclassifications.

Handling of complex defects

If LISSA does not classify an indication automatically then the classification has to be done by a user. This can imply simply accepting a suggestion given by LISSA but may also require specifying the extent of the defect and the case parameters. These tasks are somewhat cumbersome in the current prototype implementation, for example, specification of the extent of a defect is inconvenient and sometimes can be inaccurate. Also, specifying case parameters takes too much time and could be speeded up by a clever use of default values.

The added value. When used in place of existing inspection software LISSA could potentially offer the following advantages:

- faster inspection — in case of pipes with a low amount of corrosion,
- ability to analyse more defects in pipes with high-density pitting — within given time limits,
- better repeatability of inspection,
- easier to evaluate several frequency channels simultaneously,
- more detailed reports of the inspected pipes — several defect classes distinguished, more indications per pipe.

6.7.2 Different way of working using LISSA

To achieve best results when working with LISSA it may be necessary to use special settings of the acquisition hardware, different from normally used, so that various indications (defects, dents, baffles) are best separated from each other. For example, it is advisable that in the channels used for detection the phase of baffle signals is different from the calibration range phase, as this results in much better deepest-first detection.

Because the system looks at the shape of signals it is important that the signals do have some distinguishable shape instead of being almost flat lines like in the case of b* data files in channel #2. Setting the system phase and gains so as to flatten the signal makes it easier for an operator to detect anomalous signals. However, LISSA has problems with distinguishing such signals because, for instance, baffles have a phase exactly 180 degree different from pitting. More important, all flat shapes look almost the same requiring a use of very high value of threshold *T1*.

Though the LISSA prototype can work only with two frequencies, there would be no problems with adding more channels to the system. This might improve its ability to distinguish between defects, artefacts, dents, and deposits.

6.7.3 Deployment perspectives

The results of experiments with LISSA are satisfactory enough to warrant further work on the software with a commercial product as a goal. This requires, first of all, improvement of the software, mainly the user interface but also some of the algorithms used. Once the interface and the algorithms perform fully satisfactorily it would have to be integrated with data acquisition hardware and software leading to a full product that could be used for in-field EC inspection.

After the evaluation experiments it has become clear that the following software improvements are necessary to obtain better results:

- *Better phase determination* — Problems with the current phase determination algorithm have also been mentioned. Input from the future system users is necessary to define the way the phase will be determined for complex indications. In principle, it is possible to use different phase-calculation algorithms depending on the shape of a curve. For some curves, similar to the ones used for the calibration, one could calculate the phase using the rotation angle obtained from the shape matching algorithm.
- *Case-base consistency checking* — In the current LISSA version it is possible to enter cases that are inconsistent with other cases already present in the case-base. The inconsistencies can be detected afterwards by making histogram of the matches, but then it may be too late, because a large amount of data may have to be scanned anew. It would be better to have the system scan the case-base for possible inconsistencies every time a new case is entered.
- *Retrieval of several similar cases* — Currently, only one best matching case is retrieved from the case-base. If two or more cases were retrieved, and then they were compared, this could lead to more reliable results.
- *User interface* — Improvement of the user interface is possibly the most important improvement required for making LISSA suitable for field use.
- *More advanced signal preprocessing* — The results obtained with the system could be further improved if, for example, wobble noise could be reliably removed from the signal. This type of noise distorts the shape of indications, making it necessary to have several "versions" of the signal from the same type of flaw stored in the case-base. However, because this type of noise lies in the same frequency band as the defect signal, the removal problem is quite complex.

Once the above mentioned changes have been made to the software it has to be integrated with acquisition hardware in order to obtain a system that can be used to do field inspections. Afterwards, the whole system has to be tested in real inspection conditions and the feedback from the operators used to improve it. Section 5.5.4 has already mentioned that this would typically involve cooperation of three parties: the software developer, the hardware manufacturer and the inspection company. Such a cooperation is quite difficult to realise.

6.8 Conclusions

It is easy to notice that a large part of this chapter has been devoted to signal processing (like filtering, mixing) and to defect detection. This was also reflected in the development effort. The reason for this is simple — a CBR system will be only as good as the data it receives as input. The fewer irrelevant distortions are in the data, the easier the task of the CBR system in comparing the cases. These distortions can be caused by noise, by overlap of the signal from defects and construction elements (e.g., defects under baffles) and by problems in defect extension determination. A CBR system can, in principle, handle them by storing more cases in the case-base, but a better solution is to remove the distortions before the signal is passed to CBR.

The results obtained with LISSA show that CBR is a viable methodology for automatic defect interpretation in EC inspection. First of all, it gives good results. LISSA can classify, depending on the type of data, from reasonable to high percentages of indications automatically. At the same time

it is reliable due to the inherent reliability of CBR as an instance-based method, storing all data exemplars. The high reliability is also a result of the design of the system for cooperation with an operator. Because it does not attempt to classify everything it sees automatically, it is not prone to making mistakes on complex indications. Instead, it presents them to the operator.

Because of the cooperation with an operator, the system is also capable of learning to recognise new indications. No special period of training is necessary and little a-priori information is needed before the system can be used for a particular inspection. This makes it flexible and suitable for various inspection types.

The system is also fast enough as far as the algorithms used are concerned. Certainly it requires fast hardware, but nothing more than a modern high-end portable PC type computer could provide. As far as the overall speed of inspection is concerned, it depends to a large extent on a good user interface, so if the software is to be further developed much attention must be paid to this in a field-ready implementation.

An important advantage of the CBR approach is that the automatic interpretation functionality can, in principle, be added to existing EC inspection software. This means that the software can, if desired, be used in a usual way, allowing for a gradual introduction of the new inspection technique. This should increase its acceptance. LISSA should work well with the same settings (frequencies, gains, phases) as used for traditional inspection; however, better results can be obtained if the settings are chosen such as to facilitate distinguishing of various indication types. Also, LISSA is in principle capable of inspecting multiple frequency channels simultaneously which may be difficult to do for an operator. This capability could contribute to more reliable inspection.

LISSA can provide detailed reports of *all* the incidents found in the tube, not only with the wall-loss estimates but also with class descriptions. In current inspection methods this is often done only for several biggest defects, thus not describing the full condition of a tube. Moreover, when LISSA is used, each inspection gives us a case-base that contains various indication types. They could be used, for example, for operator training.

LISSA has been developed with tube inspection in mind, however, it could as well be used for other types of EC inspection, especially where the shape of the indications is important, for instance for inspecting riveted connections. It could also be useful for simultaneous analysis of more data channels, as for example from array probes. Also one could try to use CBR to solve some complex interpretation problems, such as mentioned in Section 6.1.4. This could be done, for instance, by filling the case-base with example cases for complex defect types generated from numerical models of the inspection technique. However, using simple calibration curves for characterisation of these complex signals would probably be not as easy as for the normal defects. To solve this problem, one could use CBR not only for classification (as it is done in LISSA) but also for characterisation, for example, by doing interpolation between several retrieved cases in the adaptation stage of CBR.

6.9 References

ASME (1995) *ASME Boiler & Pressure Vessel Code.*

Benoist, B., Gaillard, P., Pigeon, M., and Morizetmahoudeaux, P. (1995) "Expert system for the characterization of defect signals in steam generator tubes", *Engineering Applications of Artificial Intelligence*, Vol. 8, No. 3, June, pp. 309–318.

Bowker, K.J., Warnes, M.C., and Ashworth, M.D. (1995) "Some developments in eddy current techniques for in-service inspection", *Insight*, Vol. 37, No. 3, March, pp. 163–168.

Cecco, V.S., Van Drunen, G., and Sharp, F.L. (1983) *Eddy Current Testing*, Vol. 1, Atomic Energy of Canada Limited, Chalk River Nuclear Laboratories, Chalk River, Ontario.

Dekking, F.M. and van Otterloo, P.J. (1986) "Fourier Coding and Reconstruction of Complicated Contours", *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 16, No. 3, May/June, pp. 395–404.

van't Hoff, P. (1996) *Design of a case-base for Eddy Current CBR system: optimal case representation and retrieval*, TNO-report, FSP-RPT-960101.

van't Hoff, P., Jarmulak, J., Kerckhoffs, E.J.H., and van't Veen, P.P. (1997) "Methods for Transformation Invariant Representation and comparison of closed curves", *ASCI'97 Proceedings of the third annual conference of the Advanced School for Computing and Imaging, Heijen, the Netherlands, June 2-4*, pp. 167–173.

Jarmulak, J. (1997) *LISSA - a system for eddy-current data interpretation using case-based reasoning*, TNO report, HAI-RPT-970055.

Persoon, E. and Fu, K. (1977) "Shape Discrimination Using Fourier Descriptors", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 8, No. 3, May, pp. 388–397.

Pfisterer, H. (1985) "Kontrollkörper nach DIN 54 141 Teil 2 für die Wirbelstromprüfung von Rohren", *Materialprüfung*, Band 27, Nr. 12, Dezember, pp. 375–381.

Poetz, F., d'Annucci, F., Waas, A., and Eser, B. (1992) "Commercial Inspections and Authority Supervision A Partnership for Quality Assurance Demonstrated for Eddy Current Testing of Steam Generator Tubes", *Non Destructive Testing 92*, C. Hallai and P. Kulcsar (eds.), Elsevier, pp. 341–345.

Press, W.H., Teukolsky, S.A., Vetterling, W.T., and Flannery, B.P. (1992) *Numerical Recipes in C: The Art of Scientific Computing*, 2nd ed., Cambridge University Press.

Stepinski, T. (1990) "Analysis of eddy current patterns", *British Journal of NDT*, Vol. 32, No. 12, December, pp. 631–633.

Udpa, S.S. and Lord, W. (1984) "A Fourier descriptor classification scheme for differential probe signals", *Materials Evaluation*, Vol. 42, August, pp. 1136–1141.

Chapter 7

Application 2: Ultrasonic Rail-Inspection System

The second test case within the ANDES project concerned developing an automatic interpretation system for the Ultrasonic Rail-Inspection System (URS) owned by NS Ultrasoonbedrijf. This chapter describes results of the research done with respect to that problem. The chapter follows a line similar to that of the chapter about EC inspection. After a short introduction to the problem of railroad inspection, a description of the URS system is given with the emphasis on the kind of data it produces.

After it is clear what data is available and how it is interpreted by the operators a consideration is given to the choice of the automatic interpretation technique for the new system. The chosen technique – CBR – has been implemented in a prototype software. Interesting details of its design are described. Then the results of the tests are presented and discussed. The chapter ends with a discussion of the perspectives of deployment of the system.

7.1 Problem description

7.1.1 Rail inspection

Safe operation of railway nets requires periodic inspection and maintenance of the rail infrastructure. Large part of this inspection is visual (either directly by sight or aided with video cameras, laser scanning systems, etc.), for example:

- visual inspection of the state of the rails and rail track — external damage, surface wear, attachment to sleepers, etc. [van der Werf, 1993],
- visual inspection of the overhead wire [van Katwijk, 1996],
- inspection of the clearances to lineside structures and between passing trains (the so-called free profile).

However, visual inspection is not able to detect internal damage in the rails (e.g., cracks). These are usually inspected using ultrasonic methods.

7.1.2 Defects in the rails

Defects in the rails are caused by manufacturing faults and railroad operation. Faults due to a wrong manufacturing process can have a form of inclusions, flakes, piping and can be caused by impurities in the source slabs, faults in the rolling process, faults in the thermal treatment, too fast cooling, etc. These faults can be detected by making inspection a part of the manufacturing process [Sladojevic, 1995].

If the manufacturing faults remain undetected, the loads occurring during railroad operation may cause them to grow and lead to a damage of the rail. A typical example of such a damage is the kidney-shaped fatigue crack. It originates in the impurities or flakes present in the head of the rail and slowly grows as the rail is subject to bending loads from the train wheels. This type of defect can grow to cover the whole cross-section of the head and cause the rail to break. Other manufacturing defects caused, for instance, by wrong thermal processing of the rail may lead to long horizontal cracks in the head, and a faulty rolling process may lead to longitudinal splits of the web.

Defects can also be caused by an increased stress around the bolt holes drilled to join the rails with each other (fish-plated joints). The stress may lead to cracks developing from a bolt hole to an edge of the rail, again leading to a break of the rail. The area of fish-plated joints is, in general, sub-

ject to increased stress from wheels going from one rail to another. This may cause splitting or cracks of the rail head, or a separation of the head from the web.

Another source of defects in the rails are the welded rail connections, especially the aluminothermic (thermit) welds. These are performed by burning a mixture of aluminium and iron oxides. Aluminium reacts with the iron oxide producing heat which sustains the reaction. To obtain good quality welds a special mixture is used which when burned produces a weld metal which closely matches the welded rails. When this molten metal is poured into a mould encasing the gap and the rails a weld is formed. The whole process is difficult to precisely control and some thermit welds may have voids or incomplete melts within them. These, when subjected to loads, may lead to growing cracks and breaks.

Another defect source is corrosion which affects especially the base of the rail in the areas where the water may accumulate (like level crossings) and may lead to breaks in the base. These breaks are, however, usually longitudinal and the consequences are not as severe as breaks of the head or web. The rail is also a subject to the normal wear which leads to a damage of the top surface of the rail, for example, resulting in a "wavy" profile of the rail. This is especially the case in the bends of the rail track. The top surface of the rail can also be damaged by "burns" from spinning wheels. (More examples of these and other rail defects are given in [Bray, 1991].)

7.1.3 Ultrasonic Rail-Inspection System

Since 1986, Dutch Railways use the Ultrasonic Rail-Inspection System (URS) to inspect railway tracks [Esveld, 1989; Roos, 1990]. The rail inspection is done using a special car, Figure 7.1 shows the new version taken into service in Autumn 1997. The car has two assemblies of ultrasonic transducers (one per rail). In the first generation system the transducers were placed in an assembly at 0° and ± 70° angles (see Figure 7.2), providing information in 4 channels: 0° echo, loss of the bottom echo (BEV0), and ±70° echoes. The new, second generation system uses 4 additional channels (±70° side beams and ±35° beams) to improve defect detection (an outside view of the new transducer assembly can be seen in Figure 7.3). Measurements are made every 2 or 3 mm along the rail depending on the travel speed. Knowing the time elapsed from the send-pulse till the reception of an echo, the positions of echoes are calculated. When indications coming from a section of rail are

Figure 7.1: New ultrasonic-inspection train.

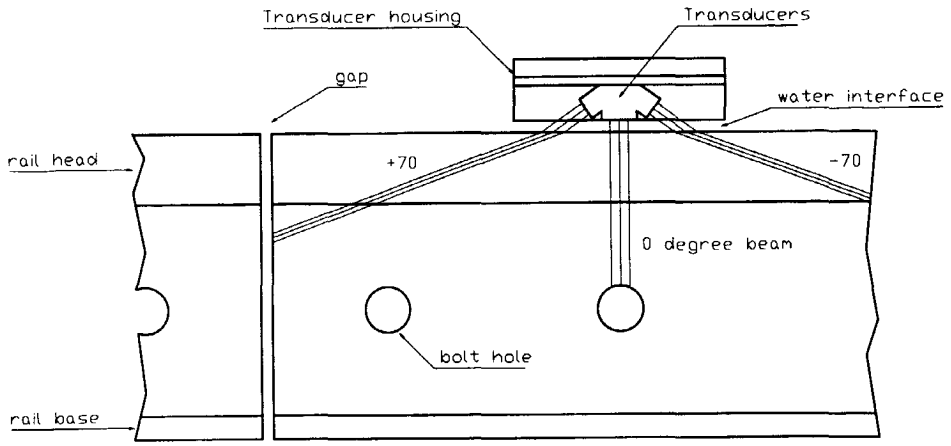


Figure 7.2: Schematic view of the ultrasonic transducer assembly (from the first generation URS) placed over one half of a fish-plated joint.

put together a B-scan, for example, as shown in Figure 7.4 is obtained. Any defects of significant size present in the rails should be visible in the images, in addition to indications from rail constructions and/or noise.

In the following sections the second generation URS^{nt} system is described. However, because the inspection train is new, no data from it was available when the data-interpretation system was being developed. This means that sections which deal with data interpretation, are based on the 4-channel data from the old URS system.

System Components. The URS^{nt} is composed of two subsystems: the incident detection system (IDS) which works on-line in real time and the report generation system (RGS) for off-line data classification and reporting, see Figure 7.5. The ultrasonic transducers placed in a special assembly

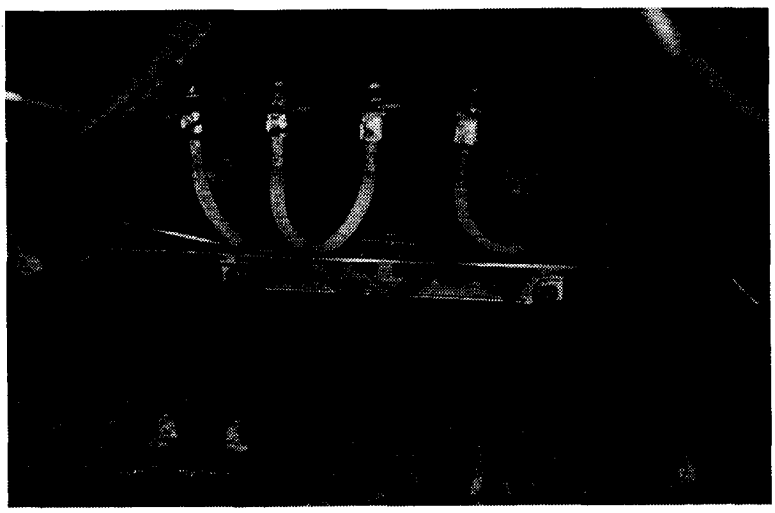


Figure 7.3: New 8-channel transducer assembly. The white hoses supply water for the water interface.

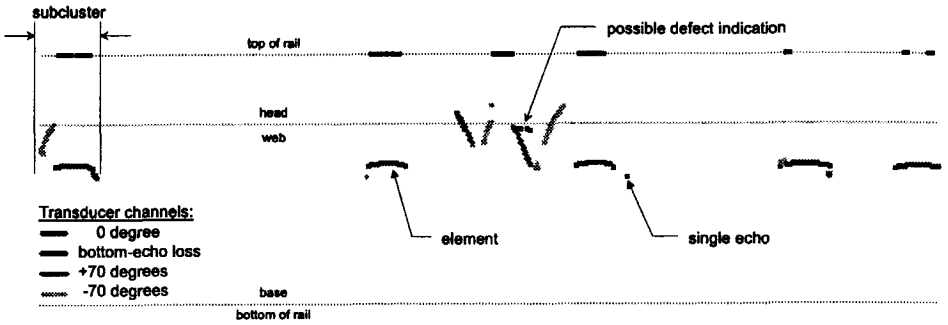


Figure 7.4: B-scan image from the URS system (vertical placement of the bottom-echo-loss indications is arbitrary). The image shows a fish-plated joint with 2+3 bolt holes. 0° indication in the joint area could indicate a possible defect.

glide on a water film over the rails. The pulser/receivers are placed on the bogie close to the transducers, the rest of the sensor electronics is placed under the carriage. The sensor electronics processes the echo signals, detects the echo peaks, and sends that information to the workstation. The signal from the transducers can be observed on an oscilloscope (see Figure 7.6) and the detected echo incidents are shown on a screen plot and plotted on a printer (see Figure 7.7). Both can be used to judge the quality of the data and based on it the settings of the sensor electronics can be adjusted. A clustering algorithm (described in [Giling, 1996]), which extracts the incidents belonging together from the data stream (these will form the individual B-scan images), is run on-line on the workstation. This algorithm also removes the isolated noise-incidents present in the original data.

RGS, see Figure 7.8, consists of computer- and interactive-classification software for B-scan image classification, and an Oracle database system for archiving and reporting the results.

Current work procedure. The train crew consists of minimum four persons: a driver, two IDS operators (a controller and an observer) and an RGS operator. During the ride the IDS operators monitor the quality of data on the scopes and on the plotter screen, see Figure 7.6 and Figure 7.7.

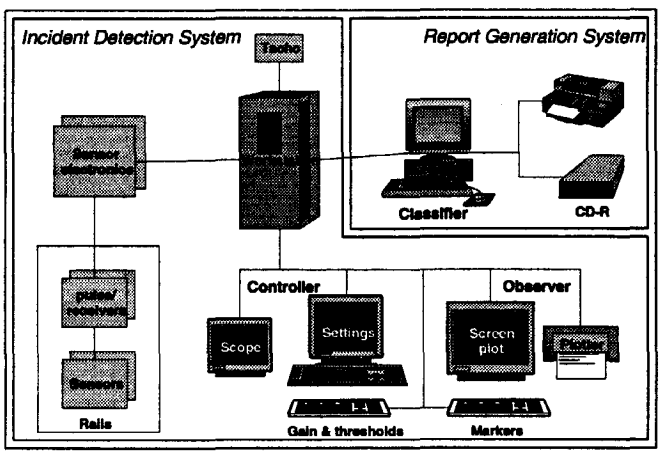


Figure 7.5: Components of URS^{nt}.

Figure 7.6: Controller console of the IDS. From left: control pulpit for the transducer assemblies (lowering onto the rails, water supply), oscilloscope, and workstation monitor with system settings.

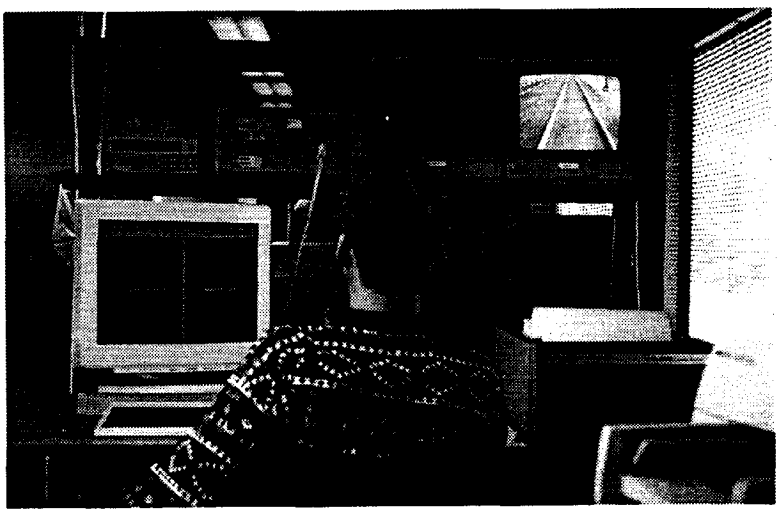


Figure 7.7: Observer console. Screen plotter on the left, paper plotter on the right. Video monitors on top show the rail track in front of and behind the train.

If necessary they may change the parameters of the echo-detection electronics. The data acquired by IDS is stored on a hard disk drive.

After a part of the track has been scanned, the data is transferred from the IDS to the RGS. The data is processed by a computer classification program RGCOMC. RGCOMC subclusters each image and then processes the results using a CLIPS expert system. The classification results are stored in a cluster file. The RGS operator then reads the data into an interactive classification program (RGINT) where he can look at the images not classified by the expert system and enter the definitive classifications, see Figure 7.8. The detected defects and question points are entered into a database and a report can be printed out.

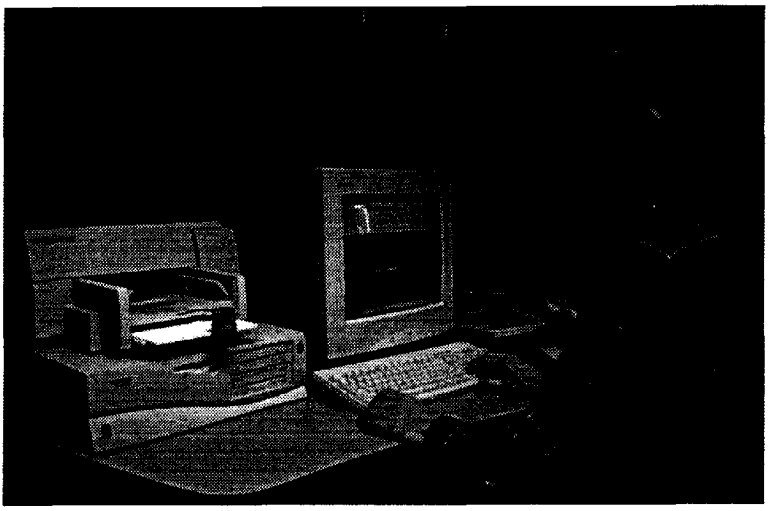


Figure 7.8: Classifier console.

Reported indication types. There are various types of B-scan images showing rail constructions, noise, defects, or combinations of these. The most common image type is the loss of the bottom echo constituting approximately 60% of all images. These are trivial to recognise. The research concentrated on the recognition of the remaining images and, whenever recognition percentages are presented here, they do not include the bottom-echo-loss images.

There are more than 30 image classes which that have classification codes defined for them (see Table 7.1). They can be divided into three groups: construction elements (OBJ), noise and distortion (STO), and (possible) defects (VFP). Some of these image classes rarely occur. Figure 7.9 lists the classes which occur most often (apart from the bottom-echo-loss – BEV) and the ones which are important to recognise (defects). As can seen the defects are rare. The so-called question points (images with possibly defects) constitute only about 1% of the non-BEV images. The low percentage of defects is common in NDT problems.

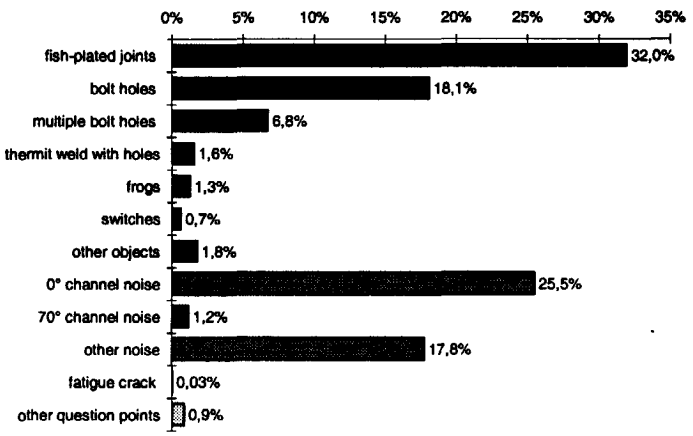


Figure 7.9: Most common B-scan image types - from the 128 data files used for testing (percentages add up to more than 100% because some categories may overlap).

Table 7.1: Image class codes used in the URS system. OBJ stands for object, STO for noise (storing), and VFP for question/defect point (vraag/foutpunt).

Code	Type	English name	Dutch name
BEV	STO	bottom-echo loss	bodemechoverlies
BG	OBJ	bolt hole	boutgat
BGO	OBJ	bolt-hole object	boutgat-object
B2	OBJ	double bolt hole	dubbel boutgat
BL	OBJ	arc weld	bekist las
BO	OBJ	bridge transition	brugovergang
CI	OBJ	expansion device	compensatie-inrichting
CL	OBJ	expansion joint	compensatie-las
LL	OBJ	glued insulated joint	lijmlas
TL	OBJ	thermit weld	thermietlas
T0 (or TL0)	OBJ	thermit weld with holes	thermietlas met boutgaten (geen voeg)
PL	OBJ	fish-plated joint	plaatlas (met wijde boring)
PL_KB	OBJ	fish-plated joint with short bolt-hole distance	plaatlas met korte boring
PSK	OBJ	frog	puntstuk
SL	OBJ	flush-butt weld	stomplas
SPS	OBJ	rail	spoorstaaf
VFP	VFP	question/defect point	vraag/foutpunt
V	VFP	question point	vraagpunt
F	VFP	defect	foutpunt
ZVB	VFP	kidney-shaped fatigue crack	zilvervlakbreuk
NTC	VFP	to be classified	nog te classificeren
S0	STO	0° noise	storing 0 graden tasters
S70	STO	70° noise	storing 70 graden tasters
W0	STO	0° water-interface disruption	storing 0 graden waterinterface
W70	STO	70° water-interface disruption	storing 70 graden waterinterface
ES	OBJ	insulated joint	elektrische scheiding
BR	OBJ	bridge	brug (oud)
OW	OBJ	level crossing	overweg
WL (or WSL)	OBJ	switch / point	wissel
KSK	OBJ	diamond crossing	kruisstuk
TG	OBJ	switch blade (tongue)	tong
OL	OBJ	resurfacing	oplassing
NLPT	OBJ	emergency fish plates	noodlasplaten

RGCOMC, the current URS^{nt} computer classification system, can recognise some images in the following classes: VFP, BG, BGO, TL, TL0, BEV, S0, W70 and the default NTC. The VFP and NTC marked clusters have to be interpreted by the operator. Though the operator can enter any of the possible codes to describe an image, because of the lack of time only VFP, V and F class codes are entered. This means that many images remain with the NTC classification even if they were seen by the operator.

7.1.4 Results of the old system

In [Roos, 1990] some statistics about results obtained by the previous URS/RGS system are reported. In the period from 5 November to 5 December 1990 approximately 800 km of track were scanned. The data was saved in 75 files. The total number of image clusters that had to be classified was 61,453 (approximately 820 per file). Among them, the following numbers of clusters were automatically classified:

- bottom-echo loss (BEV) — 33,582 (55%),
- 0° noise (S0) — 4,187 (7%),
- water interface disruption (W70) — 2,958 (5%),
- bolt holes (BG) — 2,251 (3.7%),
- fish-plated joints (PL) — 1,685 (2.7%),
- thermit welds (TL) — 3690 (6%).

This gives a total of 48,353 (79%), which means that the remaining 13,100 (21%) images had to be classified by the operator (approximately 625 per day). The BEV and TL clusters constitute usually about 60% of all the clusters. They are easy to recognise — the new software does it using two hard-coded if... then... rules. If the numbers of the recognised clusters are expressed as percentages of non-BEV clusters then the following results are obtained:

- 0° noise (S0) — 17.3%,
- water interface disruption (W70) — 12.2%,
- bolt holes (BG) — 9.3%,
- fish-plated joints (PL) — 7.0%.

This adds up to roughly 46%. Since then, the PL detection has been abandoned because it was unreliable. Some of the parameters of the system have changed so that W70 clusters are not being recognised anymore. If these two are removed this would give a 26.6% recognition ratio.

Of the total 61,453 clusters 57 were judged to be defects and 3 as question points. It should be noted that the 75 files mentioned here contained a quite high percentage of W70 clusters, they occur less often now (compare to Figure 7.9).

7.1.5 Requirements for the new system

The above-mentioned data interpretation system is sufficient for processing data from the old inspection system capable of a maximum scanning speed of 50 km/h and having 4 channels per rail. With the help of automatic classification it was possible to deliver a full inspection report at the end of each working day. The new inspection system has more transducers per rail (8 instead of 4) and a higher inspection speed (75 km/h). To keep up with the larger amount of data a better automatic classification system (achieving higher recognition ratio) had to be developed.

7.2 Problem definition

The problem that has to be solved concerns classification of images, in which pixels correspond to echoes from various channels, into two main classes: defects or non-defects. The recognised non-defect images do not have to be classified by the operator and thus the system contributes to the reduction of the workload on the operator. The possible combinations of actual rail condition (as far as it is possible to determine from the ultrasonic B-scans) and the automatic classifications are shown in Table 7.2.

In the process of recognising non-defect images, the image types (e.g., bolt hole) are determined. The main purpose of distinguishing non-defect classes is to ensure the objective reliability and the subjective acceptance of the system. Possible defect images are rare (less than 1% of all the images), therefore, defect/non-defect classification alone would be hardly sufficient to judge the reliability of the system. There are days when no defect is found and having the system return "non-defect"

as the only classification for all the images would certainly not increase the users' trust in the capabilities of the system. The more detailed classifications can also be used by the end-user for statistical analysis of the data and for operator training.

Table 7.2: Possible combinations of actual condition of the rail (as far as possible to determine from the ultrasonic B-scans) and the classifications determined by the automatic classification system.

actual condition of the rail	classification according to the system	shown to the operator	remarks
non-defect	non-defect	no	desired behaviour
non-defect	defect	yes	acceptable, but reduces confidence in the system
non-defect	unknown	yes	acceptable if not too often
defect	defect	yes	desired behaviour
defect	non-defect	no	**should not occur**
defect	unknown	yes	almost the desired behaviour

The two main parameters which describe the performance of the system are: the percentage of the images automatically classified (not shown to the operator) and the percentage of the defects classified as non-defects. The first percentage has to be as high as possible and the second one as low as possible. For the inspection company it is especially important that the percentage of missed defects is low. To keep it low one is ready to sacrifice the overall automatic recognition ratio.

A system destined for field use should be able to do the classification fast enough. In practice it means that it should be at least as fast as the human inspector. As far as the speed is concerned, a distinction can be made between average processing speed (calculated over the hole data set) and the worst-case speed (e.g., for a very complex image). The worst-case speed is important when the system has to work interactively with an operator. Ideally, the system should be fast enough for interactive operation, but batch operation (like RGCOMC) is also acceptable.

7.3 Choice of AI technique — the argumentation

The first decision that had to be made was the choice of an appropriate data interpretation technique (or techniques). As the descriptions of similar systems (which could provide ideas of techniques that do work) are rare and lack detail (e.g., [Lesiak, 1992; Havira & Chen, 1995]), the decision had to be based on the experience with the old URS and on the analysis of the general capabilities of the candidate techniques.

Problems with statistical classifiers. Early in the project it became clear that a statistical classifier, or a neural network, was not suitable for this problem, because of the following reasons:

- difficulty in representing the information about the image in a form suitable as an input for a typical statistical classifier,
- large variety in the data making the collection of a training set, as well as the design of the classifier (data representation), a difficult task,
- small percentage of the defects in the data (classes are not evenly represented),
- small differences between some defect and non-defect images (images from different classes can be very similar),
- large differences between images from the same class.

These problems would make it difficult to design a good (reliable) statistical classifier. Certainly, to achieve good results a lot of a-priori knowledge would have to be incorporated into the classifier design.

7.3.1 Rule-based classification

Many of the simpler B-scan images and even the more complex ones (if they are "ideal", i.e., with all component indications present and without noise) can be relatively easily described using rules. The most obvious example is the loss of the bottom echo (BEV) where a single CLIPS rule

```
(defrule detect-sub-bev "BEV0 only"
cluster (clas ntc))
?sc <- (sub-cluster (num ?i) (clas ntc)) ;; still to be classified
(sub-cluster-kn (num ?i) (typ bev0)) ;; there is a bev0 channel
(not (sub-cluster-kn (num ?i) (typ ~bev0))) ;; there are no other channels
=>
(modify ?sc (clas bev)) ;; subcluster classification is bev
)
```

can classify approximately 55% of all images. Other image types which can be easily described by rules are S0, BG, BGO, and T0 (though some T0 clusters can be confused with PL clusters with missing 70° echo indications). Since 1990 such rules (or SQL queries) have been used to perform automatic classification in URS.

Classification using SQL queries. The first generation URS system made use of SQL queries to an Oracle database which operated on tables of data obtained from image subclustering. The SQL queries assigned initial computer classifications to the images [Tange, 1990]. The queries were designed to detect the following:

- loss of bottom echo (BEV),
- thermit welds (TL),
- water interface disruption (W70),
- 0° noise (S0),
- bolt holes (BG), and double bolt holes (B2),
- fish-plated joints (PL).

It appears that the PL detection rule was used only for a short period of time because it could mistake defect thermit welds with bolt holes for good fish-plated joints [Roos, 1990] (the reasons for this are explained in the next section). Also, it seems that the parameters in the W70 detection rule were not updated after later changes in the system, resulting in no W70 clusters being detected anymore.

Previous knowledge-based prototype. In [Jansen, 1991] a prototype expert system is described which should be able to recognise bolt-holes (BG), fish-plated joints (PL) and thermit welds with bolt holes (T0). The system distinguishes 3 levels of image elements: the echo level (individual echoes), the elementary level (e.g., single bolt holes) and the composite level (the whole image). The rules developed in that project have not been used on the train mainly because of two reasons:

- The inability to do reliable recognition at the echo level, mainly because no information about the element orientation is available. This is especially critical for correct recognition of joints. For example, the existing subclustering algorithm provided insufficient information for distinguishing between a fish-plated-joint area and a cracked thermit weld. This is a significant deficiency because it could lead to the acceptance of defects.
- The rules were designed for "ideal" PL and T0 images, however, a large number of images significantly differ from the ideal ones. This is a less serious problem than the previous one, but it means that in order to achieve a high recognition ratio a large number of rules would have to be implemented.

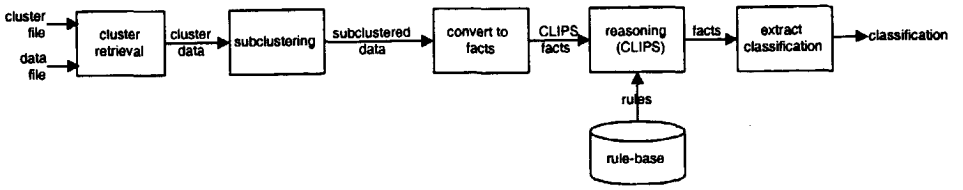


Figure 7.10: B-scan-image classification using a rule-based expert system (as implemented in RGCOMC).

Classification in URS^{nt} using CLIPS. The currently operational version of the RGS computer classification software (RGCOMC) uses a combination of the old subclustering algorithm and a rule-based system in which the rules are largely a translation of the old SQL queries. Figure 7.10 shows the outline of the system. The system uses the CLIPS expert-system shell for reasoning. It has 15 subcluster-level rules and 12 cluster-level rules. It can recognise images in the following classes: VFP, BG, BGO, TL, TL0, BEV, S0, and W70. The overall design is described in [Jarmulak, 1995] and the rules are explained in [Roos, 1995].

Lessons from the use of rules. The experience gathered using the SQL queries and the attempts to design an expert system for the first generation train show the following:

- Reliable rules for complex images are not trivial to design. Certainly not if the subclustering algorithm does not provide enough information about the image.
- Knowledge acquisition gives information mainly about typical ("ideal") images.
- Validation of the rules is necessary. Even initially "good looking" rules may turn out to be wrong in certain exceptional situations.
- Rules need maintenance. If changes are made to the system then it is necessary to check if any rules need to be updated.

7.3.2 Choice of methodology for the new classification system

Simply extending the rule-based system to classify more images would be difficult and would probably not lead to the desired results. One reason for this is that the old subclustering algorithm does not provide enough information about the image components (this is explained in more details in the next sections), therefore, a better clustering algorithm had to be implemented.

Another reason why just increasing the number of rules would not give significant improvement is the difficulty with acquiring knowledge about the images. The normal (ideal) images are relatively easy to describe. However, depending on the settings of the IDS and the scanning conditions some of the indications in the images may be missing or may be extraordinarily large. In spite of the fact that they differ from the ideal image they are classified as good images because they contain nothing which could point to the presence of defects. However, when an expert is asked how a certain image looks like he will most likely describe an ideal image; he may recall some aberrations, but certainly not all the possible ones. To classify a large percentage of images many rules would be necessary with each rule covering only a small percentage of possible cases.

Third, and maybe practically the most important, reason against simply extending the rule base is the fact that the new URS^{nt} system has 8 channels as compared to the old 4, thus the images coming from the new system are significantly different. However, when the work on the new interpretation system was done no data from the new URS^{nt} system was available. If the rules were developed for the 4-channel system, much effort would be necessary to adapt the rules to the 8-channel system.

Fourth, it was found that maintaining even a small rule base is quite difficult, and certainly too difficult to be done by the end-user of the system.

After analysing the options a choice was made to use case-based reasoning approach to the design of the new classification system. First of all, it can handle the variety and complexity of the B-scan images. But the advantage of CBR which is probably the most important for the URS is the capability of CBR systems to learn — this ability largely solves the second and third above-mentioned problems related to the use of an expert systems. Another characteristic of CBR important for URS is the work in cooperation with an operator — this fits well in the current interactive way of working with RGS. Also, a CBR system should be easier to maintain. Many other advantages of CBR as described in Section 4.4 are also of importance.

In spite of the decision to use CBR, it has been decided to retain the current rule-based classifier (though slightly modified). This way the already tested rules from the existing URS system can be used to classify a significant percentage of well defined images. Combined use of rule-based and case-based techniques has also a theoretical justification. Surma & Vanhoof [1998] present it as a means of combining and synergistically utilising general and specific knowledge, making use of a simple heuristic: *"To solve a problem, first try to use the rule-based approach. If it does not work, try to find a similar problem you have solved in the past and adapt the old solution to the new situation."*

As Figure 7.11 shows the new system consists of three main stages: preprocessing, rule-based classification and CBR classification. The preprocessing includes the new image-clustering algorithm described in the next section. Rule-based classification is largely similar to the one used in the current system, as described in Section 7.3.1. The data that the rule-based classifier uses is similar to the data used by the RGCOMC classifier. Two important differences are:

- The subclusters are the same as in the *new* clustering algorithm.
- For each of the channels a parabola is fitted and the angles of the tangents to the parabola ends as well as the error of the fit are recorded. The data for individual channel elements is not used, because the currently used rules do not require it. It may be necessary to include it if, in the future, more complex rules are added.

Altogether, the main difference between these rules and the rules currently used in RGCOMC is the use of orientation information (the angles) which should result in a more reliable bolt-hole detection. The system can detect S0, W70 (theoretically), BG, BGO and TL0. BEV and TL are detected during preprocessing.

The images not classified by the rule-based classifier are processed by the CBR subsystem described in Section 7.5.

7.4 Data and image processing

The raw data from the IDS part of the system consist of a series of echo incidents. Each incident contains information about channel type (BEV0, 0°, ± 70°, etc.), the transducer position and time to the echo reception. Knowing the physical properties of the scanning system this data can be transformed (normalised) into a B-scan image and shown on a display (Figure 7.4). Interpretation of a continuous B-scan image would be cumbersome (mainly because of large sections of rails with

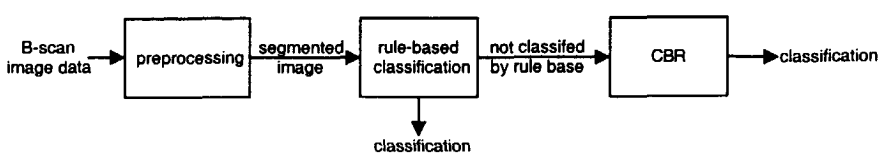


Figure 7.11: Three stages of the new hybrid classification system.

no indications) so the data has to be divided into meaningful smaller elements. These elements will correspond to whole construction elements in the rails, like a fish-plated joint for example.

The algorithm which does this clustering in the old system is described in [Tange, 1990] and the newer on-line version is described in [Giling, 1996]. The clustering is done based on the horizontal positions of incidents. Incidents lying within some threshold distance from one another are grouped together. The so-called start condition for a cluster requires presence of at least 3 consecutive incidents. Thus noise consisting of 1 and 2 incident echo points not lying within any cluster is ignored which contributes to data reduction.

This type of segmentation is sufficient if the classification is done only by the operator. For automatic interpretation, however, more information is necessary about the elements present in the B-scan images. This information can be obtained by clustering the image into smaller subclusters and even into individual elements like 0° echo from a bolt hole.

7.4.1 Old image clustering

The algorithm currently used in URS^{nt} for (sub)clustering B-scan images consists of two stages:

- first, the channels are subclustered individually using the not-normalised horizontal incident positions,
- then, the subclusters found in various channels are grouped together if their mid-positions are within certain threshold distance from one another (this method of channel combining was developed with PL images in mind and for other image types results sometimes in overlapping subclusters).

Apart from that, the algorithm attempts to fit bolt-holes to 0° echo incidents using a simple wave propagation model. For each group of more than three 0° incidents, having a length smaller than some threshold, the algorithm calculates the bolt-hole centre position and the fit error. Thus, this clustering algorithm returns the following data about the image subclusters:

- horizontal start position and length,
- percentage present (for BEV0, 0°) — this is the ratio of the number of incidents to the cluster length,
- minimum and maximum depth (except for BEV0),
- bolt-hole coordinates and fit error (for some 0°).

It is obvious that in situations when an image contains, for instance, two distinct echo lines close to each other the algorithm still returns a single subcluster. Moreover, the algorithm provides no information about the orientation of indications. However, when an operator analyses the image he certainly would take the presence of two distinct lines and their orientation into account. These deficiencies limit the capabilities of any automatic interpretation algorithm which would use data from this subclustering algorithm, see, for example, Section 7.3.1.

7.4.2 New image clustering

The new image clustering algorithm, developed as part of the ANDES project, is capable of extracting information about individual lines in an image.

Several methods of image subclustering have been tried out. The Hough transform [Gonzalez & Woods, 1993; Hopgood et al., 1993] (see Section 5.3.3), theoretically good at finding line-clusters, has been implemented but did not give good results. This is caused by the fact that many incident clusters from $\pm$ 70° transducers are not straight lines but are slightly curved. Making the polar-coordinate bins of the clustering algorithm "wider" does not help, because then close parallel lines get combined.

Fuzzy clustering algorithms, like described in [Krishnapuram & Freg, 1992], though theoretically capable of achieving good results, require a lot of memory storage and long processing, therefore, they do not seem to be the best candidates for URS.

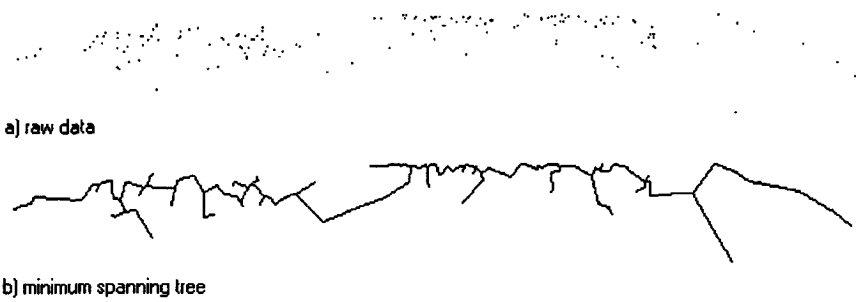


Figure 7.12: A minimum spanning tree for a noise-like group of points. MST algorithms are not the best candidates for clustering this type of data.

As described in [Jarmulak, 1996] a minimum-spanning-tree (MST) clustering has been chosen as the basis for the image-clustering algorithm. MST is a graph connecting all the vertices (in our case echo incidents) in such way that the sum of the length of all the edges of the graph is minimal. For example, Figure 7.12 shows an MST fitted through a noise-like group of points. After a minimum spanning tree has been constructed for each of the channels, it is split into clusters on the long connections. Through each of the resulting clusters, lines or parabolas are fitted, combining the clusters if possible. Where no good line or parabola fit is possible, clusters of noise are constructed.

The algorithm gives good results, except for large W70 noise images, like in Figure 7.12, where the clustering time is too long. However, such images are not very common (see Figure 7.9) and when the clustering takes too much time it is broken off and the image can be shown to the operator for classification. Figure 7.13 shows an example of clustering results for a fish-plated joint. Each of the clusters found can be either a line, a parabola or noise. For each cluster the location, the sizes, and the dispersion (the average distance between cluster points) are stored. For lines and parabolas the equation parameters are known. Additionally, for 0° clusters bolt-hole fitting is done like in the old algorithm. As can be seen in Figure 7.13 the image is also divided into subclusters but using a simpler algorithm than the one described in Section 7.4.1. This image subclustering is done on all the channels simultaneously (and not per channel and then combined).

7.5 Case-based reasoning

The motivation for the use of CBR for URS data interpretation has already been given in Section 7.3.2. This section describes the overall design of the CBR part of the system and shows how it fits into the whole. Details of some data representations and algorithms used in the CBR system are also described in this section.

7.5.1 Overall system design

The overall design of the system is illustrated in Figure 7.14 & Figure 7.15. The system consists of three stages (as already outlined in Figure 7.11). In the first stage, data preprocessing is done. This stage gets as input a single B-scan image cluster. A simple rule detects the BEV0 (and TL) clusters — usually 60% of all the clusters fall into this category. The remaining images are segmented using the new minimum-spanning-tree based algorithm.

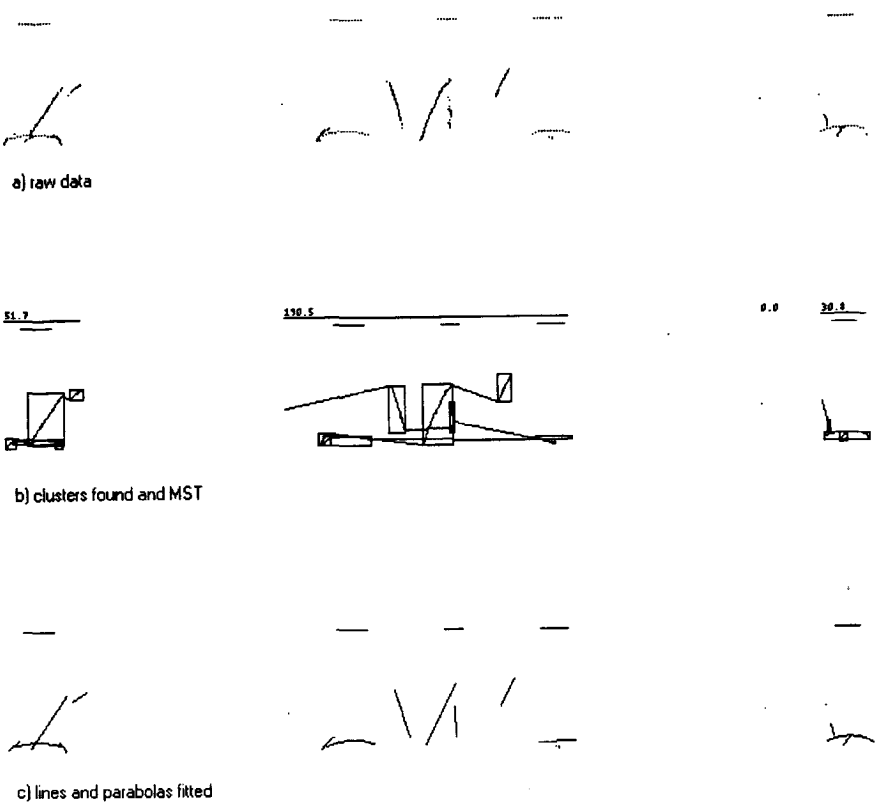


Figure 7.13: Clustering of an image of a fish-plated joint.

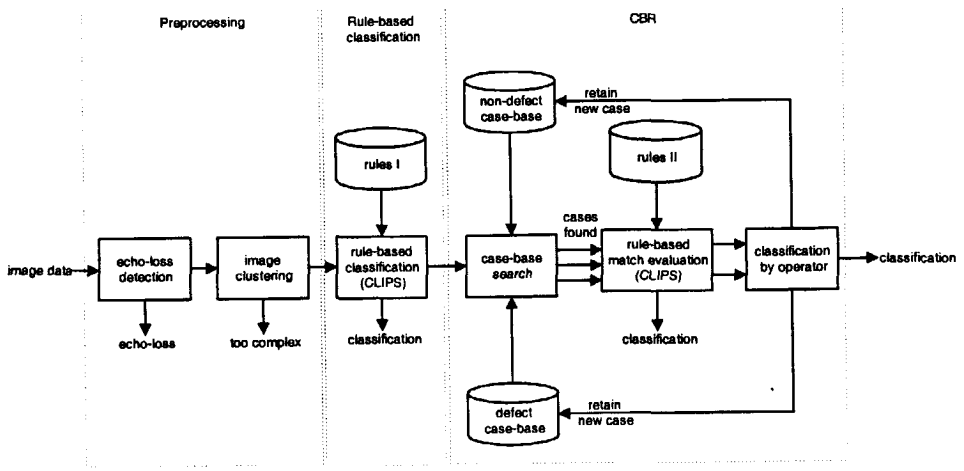


Figure 7.14: Block schema of the prototype hybrid classification system.

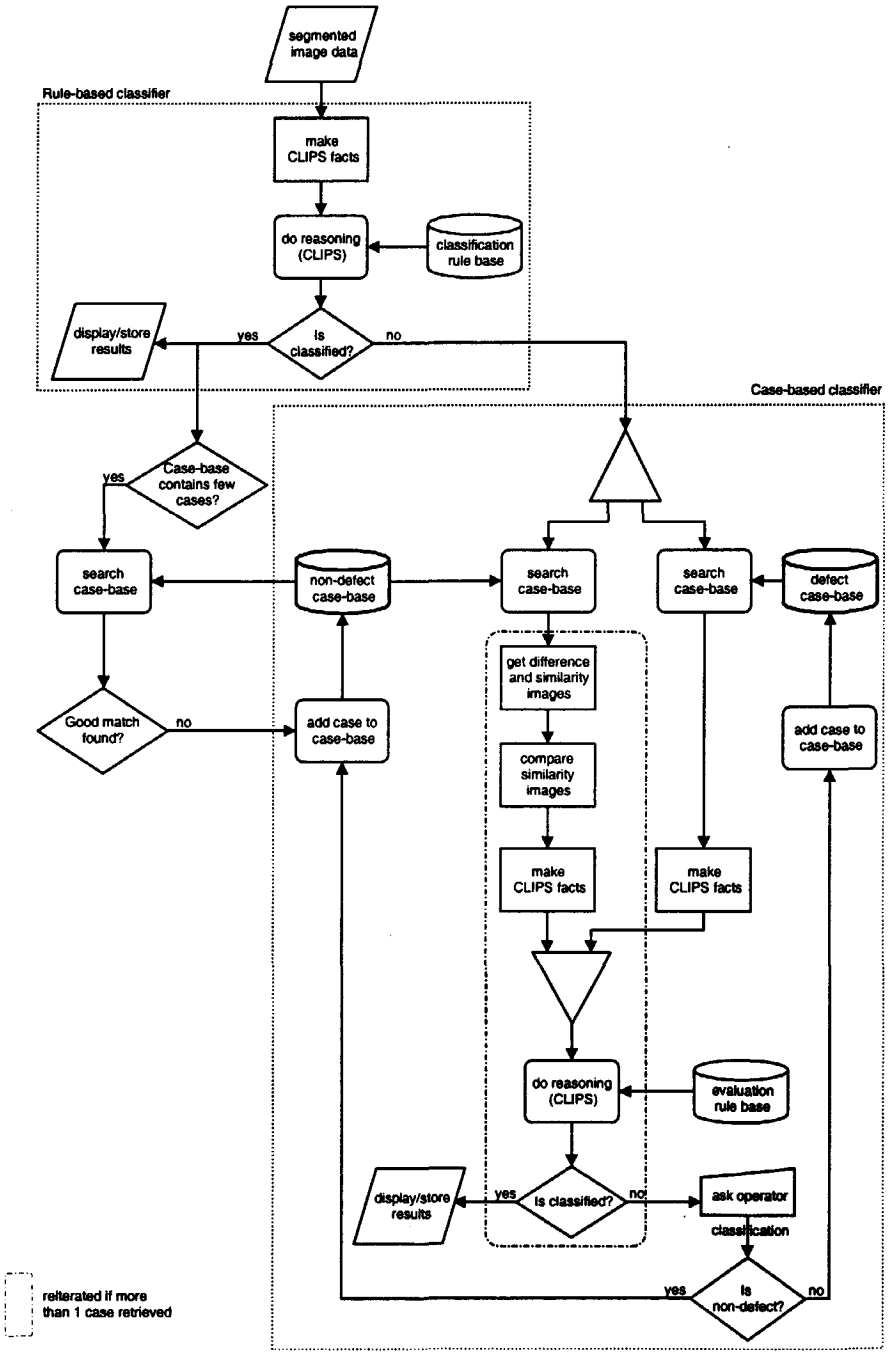


Figure 7.15: Flow schema of the prototype hybrid classification system.

The next stage does the rule-based classification (as described in Section 7.3.2) using the *rules I* rule-base. The specific reasons for the use of a rule-based classifier are:

- It is faster than the case-based classifier (for the images that it can recognise). If certain image class can be described using rules then matching against the few rules is faster that matching against a larger number of cases which would be necessary to represent the same concept. Examples of such classes are a bolt hole and 0° noise. As the results of tests (Section 7.6.3) show, the hybrid system is more than 20% faster than the case-based-only system.
- It makes it possible to use some of the available a-priori knowledge to directly classify data. For example, it is well-known what a typical bolt-hole image should look like.
- It can classify noise images, like water-interface disruption, which are difficult to classify using the case-based classifier. Therefore, the hybrid system recognises more images than the case-based-only system (again see Section 7.6.3).
- Even if it does not recognise the whole image, it can still recognise the individual subclusters — this information, in turn, can be used by the case-based classifier. In the current prototype, information about bolt-hole subclusters is used.

The remaining unrecognised clusters are processed by the CBR system. It stores images that were classified by the operator in two case-bases: one for images without defects and one for images with defects. Two case-bases are used because of two main reasons:

- Because of the required reliability the system always tries to retrieve one defect case and one non-defect case, and then the results of matches are analysed. From the implementation point of view, searching in two case bases is easier than searching in one case-base containing both defect and non-defect images. Moreover, the search can be done in parallel, in two separate search threads.
- Defect cases are rare and require time to accumulate, therefore, all the available defect cases will be used for classification. Non-defect cases are track specific, for instance, there is difference between data from inspection in Switzerland and in Holland. Therefore, it can be expected that different non-defect case-bases will be used depending on the track to be inspected. Having two different case-bases makes them easier to handle.

When a new image is presented to the CBR system, the system searches for the most similar images in both case-bases and the most similar cases with defect and non-defect images are retrieved. For the non-defect case, two similarity and two difference images are then constructed. (Given images A and B, $A \cap B$, $B \cap A$, $A-(A \cap B)$, and $B-(B \cap A)$ are obtained.[1]) Information about these images and about the results of the matches is converted into expert-system facts and sent to the reasoner (CLIPS). A set of rules is used to evaluate the differences. If the differences are not significant then the current image is given the classification of the retrieved case — there is no adaptation. The rules are also used to spot possible inconsistencies between the defect and non-defect case-bases. If no classification can be determined then the image has to be classified by the operator. In general, all the images classified by the operator are stored in the case-bases (they can be removed later during case-base maintenance, based, for example, on retrieval statistics).

Thus, two expert systems with two rule bases are used (see Figure 7.15). *Rules I* are for the classification of the simpler (typical) images. *Rules II* are for the evaluation of the match between the current image and the image(s) retrieved from the case-base(s).

1. $A \cap B$ denotes all the elements of A that match the elements of B, and $A-B$ denotes removal from A of all the elements that match with the elements of B. Note that the notation is only analogical to set notation and $A \cap B$ is not the same as $B \cap A$.

The part of the schema in Figure 7.15 outside of the stippled rectangles represents the routines responsible for filling the case-base with prototypical images. If this feature was not present then, for example, only bolt-hole images deviating from the normal would be present in the case-base because the rule-based classifier recognises all the normal bolt holes.

7.5.2 Implementation

The prototype has been entirely written in C++ and apart from CLIPS expert-system shell (version 6.03) no other tools were used. The CLIPS code has been embedded in a server application called Clipsor (CLIPS processor). Clipsor can accept data facts (via network) from a client application, process it and send the results back to the client. Because the overall system uses two rule-bases, two Clipsor servers are used. This means that the running system consists of three processes. They can be run on several computers to increase the processing speed and also to simplify the rule debugging (more information is visible simultaneously).

7.5.3 Case description

A single case contains information obtained from a B-scan image using the MST-based clustering algorithm plus the classification belonging to the image. The information is stored in a hierarchical structure with levels containing information about: the whole image, the subclusters, the channels, and the elements:

- At the lowest level is the element data describing individual lines, parabolas, or regions of noise. An element contains the following data, see Figure 7.16:
  - element type — line, parabola, or noise,
  - line/parabola equation,
  - centre — x,y coordinate,
  - bounding-rectangle coordinates,
  - number of incidents,
  - dispersion — average distance between incidents (echoes),
  - bolt-hole centre co-ordinates and the fit error (for F0 channel if a bolt hole was fitted),
  - coordinates of all incidents — used only for display and not for matching, stored in a separate data file.

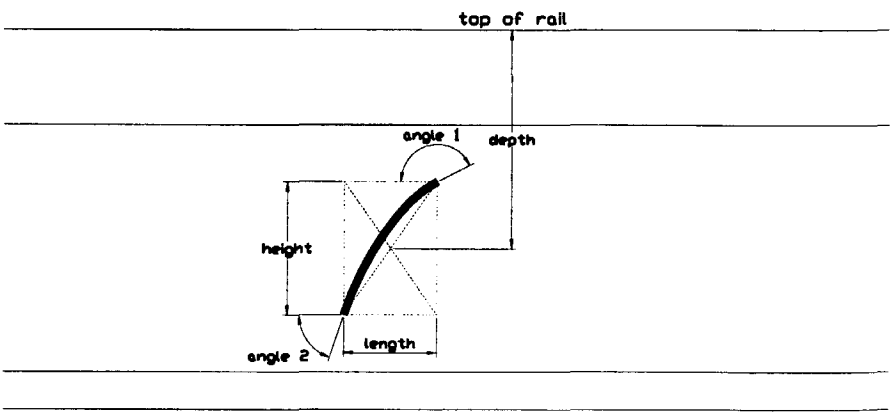


Figure 7.16: Some of the image element parameters.

- The channels contain all the elements belonging to the channel.
- The subclusters contain the following information:
  - all the channels in the subcluster,
  - centre position (horizontal only),
  - class assigned to the subcluster by the rule-based classification.
- Finally, the whole cluster contains the following information:
  - all the subclusters in the cluster,
  - classification from the operator,
  - filename and cluster number in the original data file.

For each case retrieval statistics are stored (a number of all matches done with the case as well as the number of better matches). These values are used for sorting during retrieval and for case-base pruning. Other parameters stored are the last-retrieved date and a pointer to the full original image data. At the moment no context data, like, for example, classifications of the neighbouring images, is stored in the case, though this type of data may be added in the future.

7.5.4 Case matching

A large part of the a-priori knowledge about image classification is contained in the image matching algorithm (a-priori knowledge can be also found in the image segmentation algorithm and, obviously, in the two rule-bases). Given some non-defect image the matching algorithm effectively defines a range of images which can be safely considered as non-defect (unless they are also similar to some cases from the defect-image case-base). Because an image has a four level hierarchical structure the matching algorithm also has four levels.

Element matching. At the lowest level the individual image elements are matched. The main parameters being matched are: the location of the centre point, the orientation (defined by the tangent angles at the endpoints), and the size. The differences between the elements are mapped via fuzzy membership functions into 0...1 interval as illustrated in Figure 7.17.

For several element parameters two membership functions are used: one for the absolute difference and one for the relative difference. In such cases the values of both membership functions are combined using the *maximum* as the fuzzy OR operator.

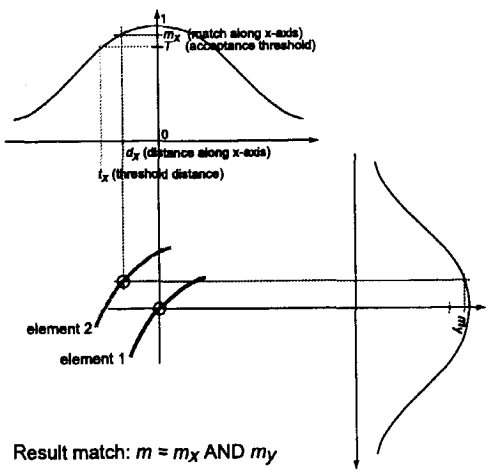


Figure 7.17: Using fuzzy membership functions to match the location of image elements.

In some situations, for example for noise elements, the membership functions are modified using a hedge, for example, a square-root function which makes the membership function "wider" but maintains the 0...1 range. Some of the membership functions depend on the value of a third parameter, for example, the allowed difference in the element location depends on the element length.

Results of the individual matches are combined using the *minimum* as the fuzzy AND operator. One advantage of the minimum is that it makes it easy to determine the element responsible for the value of the match. This has been found useful during debugging and fine-tuning of the membership functions. Another advantage of the minimum is that, especially for complex images, the result of combining all the subresults is intuitively more correct.

The sequence of steps of the element-matching algorithm is (depending on channel type not all the steps may be performed):

1. If 0° elements are being matched and through both elements bolt holes have been fitted then the matches of the fit error and bolt-hole depth are calculated (absolute difference is used), the matching it then finished. If only one element has a bolt-hole fit then the total element match will be penalised by a small factor.
2. The lengths of elements are compared. Both absolute difference and relative length ratio are taken into account.
3. The absolute difference in dispersion is compared. If BEV0 elements are being compared then comparing is finished after this step.
4. The heights of elements are compared. Both absolute difference and relative length ratio are taken into account.
5. The depths of elements are compared. Again, both the absolute depth difference and depth difference relative to the height of elements are taken into account.
6. For elements with more than 1 incident the angles of the tangents to the ends of parabolas fitted through them are compared. The allowed angle difference depends on the number of incidents, element length, and on whether the elements are noise or one of them has a bolt-hole fit.
7. All of these partial results are combined using AND operator.
8. Finally, if both elements are noise then the matching result is increased. If applicable, the bolt-hole penalty is applied (see step 1).

The element matching can be broken off if any sub-result is worse than the best match so far, this speeds-up the overall searching.

Channel matching. When matching channels within a subcluster various combinations of image elements are possible, therefore, matching is done using a graph matching algorithm where the matches at vertices correspond to the matches of the elements and the matches on the edges correspond to the mutual distance matches. The algorithm allows for an unequal number of elements but because of the speed constraints the allowed difference is limited to one element. (If the difference is higher then the match will be equal 0.0.) When matching unequal number of elements the best match is penalised by a factor proportional to the size of the left-out element.

If the extra element is omitted then the remaining n elements can be matched in $n!$ ways. For each combination of the elements the following are compared:

- The mutual positions of the elements (lengths of the edges of the two graphs formed by the elements); the horizontal distances are less important then the vertical ones.
- Matches between pairs of elements are calculated and combined using the AND operator.
- Both results are combined and a penalising factor is applied if applicable.

Not all the element combinations have to be matched — if it becomes clear the match will be worse than the best match so far then matching will be broken off. The best result for all the possible element combinations is returned as the result of matching the channels.

Subcluster matching. The subclusters to be matched should have at least one matching channel present. For each channel in the subcluster the steps of the matching algorithm are:

1. If the number of elements in a channel differs by more than one then the matching is broken off; the match returns 0.0.
2. If a channel has the same number of elements in both subclusters then the subcluster channels are simply matched.
3. If a channel has one extra element then all the possibilities of skipping a element are tried. For each of the possibilities also the distances between the locations of the centres of the subcluster channels (corrected for any missing elements) are matched — the vertical distances are more important than the horizontal ones. The best result of all the matching possibilities is returned as the subcluster channel match result.
4. If a whole channel is missing then a missing penalty for the existing extra channel will be calculated.

If both subclusters being matched have received bolt-hole classifications from the rule-based classifier then the system returns a 1.0 match. This means that is such a case no channel (and no element) matching is done.

Whole image matching. Whole B-scan images are only compared if the number of subclusters differs by not more than one.

- If there is the same number of subclusters then the pairs of subclusters are simply compared and the result is obtained by combining using the AND operator.
- If there is an extra subcluster then all the possibilities of skipping a subcluster are tried (additionally penalised by a factor proportional to the size of the extra subcluster) and the best match found is the result.
- Additionally, the distances between the centres of subclusters are compared.

Matching is computationally intensive, therefore, breaking off a matching process in progress is used extensively to speed up the overall search. Because a *minimum* is used as the AND operator for combining the subresults, it is relatively easy to determine if it makes sense to proceed with matching or not.

7.5.5 Case-base organisation

In the tests described in Section 7.6 the non-defect-image case-base grew to about 13000 cases (15.5MB). The current estimate is that the case-base will not grow bigger than approximately 50000 cases. Case-base size will be kept limited by removing seldom retrieved cases and using different case-bases for different rail-track types. Anyway, searching a case-base of 50000 cases requires efficient algorithms as well as fast hardware, especially, as the matching algorithm is quite computation intensive.

When searching, the main goal is, first of all, to find a match good enough to enable automatic classification. However, cases with slightly worse matches can still be used as suggestions for possible classifications, thus speeding-up the interactive classification by the operator. Therefore, the case-base is searched for a best match of any value.

Common techniques for speeding-up the retrieval concentrate on the reduction of the number of cases that have to be matched. In URS CBR system the emphasis is on the prioritisation of matching, so that the cases having greater probability of giving a higher match are matched first. This results in faster retrieval mainly because individual case matching can be broken-off when it becomes clear that the match will be worse than some best match so far.

In the first prototype version [Jarmulak et al., 1997] the case-base had a simple fixed tree organisation with subdivisions based on the number of subclusters, the channels present (images may have echoes from some channels missing), and the image length (the length intervals were chosen

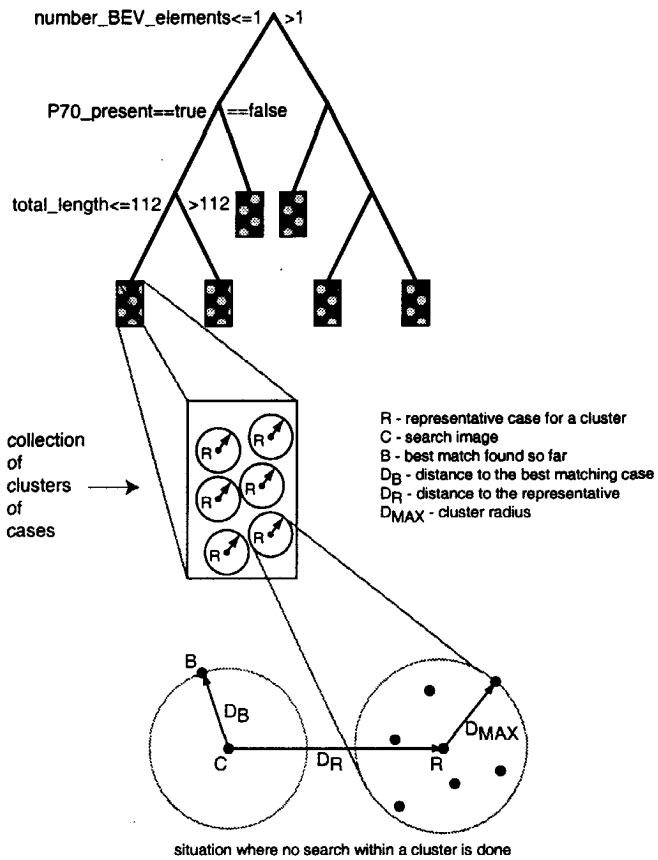


Figure 7.18: Case-base organisation (see text for explanation).

based on the analysis of the length distribution for the various image classes). The cases in the leaves of the tree were stored in simple arrays.

To improve the search speed it was decided to improve this organisation. The top level tree was built using a tree-induction algorithm, possibly leading to a more optimal form, as tree induction has the advantage that it automatically chooses the attributes which have the best class-distinguishing capability. The cases in the leaves were organised in clusters of similar cases; this organisation is used to speed up the retrieval. The new organisation of the case-base is shown in Figure 7.18.

Top-level case-base organisation. The top level of the case-base is organised in a tree. The tree is induced from data using the C4.5 decision-tree induction algorithm [Quinlan, 1993]. (C4.5 has been chosen because it can handle binary, continuous, as well as missing attributes, is easily available and simple to use.) Use of the tree can speed up the retrieval because of the fact that images from the same class usually have a similar overall look. Therefore, if images belonging to the same class are matched there is a higher probability that the match value will be high.

However, C4.5 alone would not be enough for reliable classification because there are many examples where globally similar images can be assigned to different classes (e.g., a fish plated joint with and without defect) and images from the same class can look different (e.g., a 4- and a 6-bolt-hole fish-plated joint). Of course, it is possible to make the classes finer (e.g., distinguishing be-

tween 4- and 6-hole fish-plated joints) but it still would not help in reliable defect detection, moreover, it is preferable not to change the already existing classification scheme.

For each image a number of simple parameters can be calculated including presence of echoes in a certain channel, total length, number of subclusters, number of elements in a channel, minimum/maximum depth per channel, etc. From the parameters that are relatively easy to calculate, ones are chosen that can be expected to distinguish well between the classes and then used to induce a decision tree.

Once a satisfactory (as far as the classification error and the size are concerned) decision tree has been generated, a skeleton of the case-base can be constructed on the basis of that tree. Currently, the top-level case-base structure is fixed when creating a new case-base using a tree induced using the test data set. When large case-bases are available containing data from real scans it may make sense to reorganise the case-base structure (induce a new tree) based on the data already contained in the case-base, thus optimising its organisation.

A test has been done to compare the C4.5 classification error for a tree built using the same image attributes as used in the fixed hierarchical case-base organisation and the error for a tree built with more attributes. The test was done using 13992 images to induce a decision tree, and 4664 to test it. The first tree was induced using six attributes: number of subclusters, channel presence (x4), and image length. The second tree was induced using additional 42 attributes describing the image. The results are in Table 7.3 and a part of the tree is shown in Figure 7.19.

Use of more attributes leads to better results both on the training data and the test data. Still, the difference is not large. In the 6-attribute tree 83% accesses lead to the correct category which is not bad compared to 89% for the 48-attribute tree.

```
number_BEV_noise_elements <= 1 :
|   number_BEV_noise_elements <= 0 :
|   |   F0_elements_in_biggest_sub_clust <= 0 :
|   |   |   total_number_incidents <= 31 : OBJ_SPS_STO_OTH_VFP_NON (305.0/44.8)
|   |   |   total_number_incidents > 31 : OBJ_SPS_STO_W70_VFP_NON (35.0/7.2)
|   |   F0_elements_in_biggest_sub_clust > 0 :
|   |   |   P70_present = yes:
|   |   |   |   number_F0_noise_elements <= 0 :
|   |   |   |   |   total_length <= 112 : OBJ_BG_STO_NON_VFP_NON (111.0/9.5)
|   |   |   |   |   total_length > 112 : OBJ_B2_STO_NON_VFP_NON (24.0/14.2)
|   |   |   |   number_F0_noise_elements > 0 :
|   |   |   |   |   avg_M70_element_size <= 4 : OBJ_SPS_STO_OTH_VFP_NON (37.0/6.1)
|   |   |   |   |   avg_M70_element_size > 4 : OBJ_SPS_STO_W70_VFP_NON (20.0/9.0)
|   |   |   P70_present = no:
|   |   |   |   avg_F0_element_size <= 10 : OBJ_SPS_STO_S0_VFP_NON (3876.0/37.7)
|   |   |   |   avg_F0_element_size > 10 :
|   |   |   |   |   avg_total_element_size > 143 : OBJ_PSK_STO_NON_VFP_NON (25.0/3.7)
|   |   |   |   |   avg_total_element_size <= 143 :
|   |   |   |   |   |   min_F0_depth <= 58 : OBJ_SPS_STO_S0_VFP_NON (123.0/3.8)
|   |   |   |   |   |   min_F0_depth > 58 :
|   |   |   |   |   |   |   max_total_depth <= 80 : OBJ_BG_STO_NON_VFP_NON (108.0/16.1)
|   |   |   |   |   |   |   max_total_depth > 80 : OBJ_SPS_STO_S0_VFP_NON (20.0/7.0)
|   number_BEV_noise_elements > 0 :
|   |   max_total_depth <= 57 : OBJ_SPS_STO_OTH_VFP_NON (498.0/41.8)
|   |   max_total_depth > 57 :
|   |   |   total_length <= 101 :
|   |   |   |   max_total_depth <= 86 : OBJ_BG_STO_NON_VFP_NON (2621.0/76.5)
|   |   |   |   max_total_depth > 86 : OBJ_SPS_STO_OTH_VFP_NON (29.0/1.4)
|   |   |   total_length > 101 :
|   |   |   |   avg_F0_element_size <= 37 :
|   |   |   |   |   number_fitted_BG <= 0 :
|   |   |   |   |   |   avg_P70_element_size <= 3 : OBJ_SPS_STO_OTH_VFP_NON (130.0/11.8)
|   |   |   |   |   |   avg_P70_element_size > 3 : OBJ_OTH_STO_OTH_VFP_NON (23.0/19.7)
|   |   |   |   |   number_fitted_BG > 0 :
|   |   |   |   |   |   total_length <= 336 :
|   |   |   |   |   |   |   longest_BEV_element <= 19 : OBJ_B2_STO_NON_VFP_NON (57.0/13.7)
|   |   |   |   |   |   |   longest_BEV_element > 19 : OBJ_BG_STO_S0_VFP_NON (21.0/8.0)
|   |   |   |   |   |   total_length > 336 :
|   |   |   |   |   |   |   longest_BEV_element <= 21 : OBJ_PL_STO_NON_VFP_NON (29.0/8.2)
|   |   |   |   |   |   |   longest_BEV_element > 21 : OBJ_PL_STO_BEV_VFP_NON (27.0/18.2)
|   |   |   |   avg_F0_element_size > 37 :
|   |   |   |   |   total_number_incidents <= 584 : OBJ_PSK_STO_NON_VFP_NON (28.0/9.2)
|   |   |   |   |   total_number_incidents > 584 : OBJ_PSK_STO_BEV_VFP_NON (22.0/8.1)
number_BEV_noise_elements > 1 :
|
```

Figure 7.19: Part of the decision tree built using 48 attributes.

Table 7.3: C4.5 classification errors for decision trees built with 6 and 48 attributes.

6 attributes	48 attributes
Evaluation on training data (13992 items): Before Pruning / After Pruning Size Errors Size Errors Estimate 117 2338(16.7%) 97 2337(16.7%) (18.0%) Evaluation on test data (4664 items): Before Pruning / After Pruning Size Errors Size Errors Estimate 117 811(17.4%) 97 803(17.2%) (18.0%)	Evaluation on training data (13992 items): Before Pruning / After Pruning Size Errors Size Errors Estimate 165 1359(9.7%) 127 1366(9.8%) (11.1%) Evaluation on test data (4664 items): Before Pruning / After Pruning Size Errors Size Errors Estimate 165 522(11.2%) 127 512(11.0%) (11.1%)

The errors of the C4.5 trees do not say everything about the expected case-based classification errors as the cases in the leaves still have to be matched. However, they tell something about the chance that the retrieval will find a leaf containing similar images right at the beginning of the search. This means that a smaller-error tree should result in faster overall search times, unless it is outweighed by the complexity of navigating the tree.

Bottom-level clustered case-base organisation. The leaves of the tree do not contain single-class images. That is also the reason why classification of the images cannot be done reliably using the decision tree alone (many leaves labelled as object or noise might also contain defects). When classifying a new image, images in the leaves have to be matched against it to find the closest match which may lead to a definitive classification. To speed up the search in the leaves, the cases stored in them are organised into clusters of similar cases.

The basic idea behind the clustered organisation is the trivial implication of the assumption of the regularity of the world [Kolodner, 1996] namely that if A is similar to B, B is similar to C then A is similar to C. If the similarities are calculated as distances between feature vectors in a numerical feature space, then the lower and the upper bounds for the similarity of A and C can be given using the triangle relation. For B-scan images this would be more difficult (if not impossible) but, nevertheless, case similarities can be used to organise the case-base.

Cases in a leaf are grouped into clusters of similar cases. The clusters are constructed by always adding a new case to the cluster where the best matching case has been found (unless the match is zero, then a new cluster is started). When a cluster reaches some predefined maximum size it is split (size of the cluster is limited by the number of cases not by the cluster diameter) and two new clusters are initialised with the two most mutually distant cases from the old cluster (the distance is defined here arbitrarily as *1−match*). The cases from the old cluster are split between the two new clusters — iteratively cases closest to each of the new clusters are found and the one having the smaller distance is added to the corresponding cluster.

For each cluster a "representative" (prototypical) case (R) is found, defined as the one which has the smallest maximal distance (D_{MAX}) to all the other cases in the cluster, see Figure 7.18. The clustered organisation can be used during the retrieval to direct the search first to the clusters which are more similar to the current image, as well as to skip searching too dissimilar clusters.

7.5.6 Case retrieval

The retrieval of a case that best matches some image C begins at the top of the case-base hierarchy. Based on the values of the image attributes a leaf of the indexing tree is reached. In the leaf the clusters are reordered based on the "prototypicality" of their representatives (calculated as the ratio of the number matches above a predefined threshold to the total number of matches), so that the cases which usually give better matches are matched first. After reordering, matches with all cluster representatives are calculated (distances D_R, see Figure 7.18). If no match good enough has been found, then the clusters are again reordered by sorting on the values of the calculated D_R distances and a search is performed *in* the clusters. The sorting makes it more probable that a good match

will be found in one of the first searched clusters. Given now the best match found so far (D_B) it is assumed that it makes sense to search further in any cluster i if $D_B + D_{MAXi} > D_{Ri}$. (Anyway, no searching is done in a cluster if the match with the representative is 0.0.) If $D_{Ri} > D_{MAXi}$ (C "outside" the cluster) then the cluster is fully searched without any case ordering, otherwise (C is "inside" the cluster), use is made of the fact that the average distance from C to cases in the cluster is greater than D_R, therefore, first the cases within D_R radius from R are matched and then the remaining cases.

As already said, the calculated distances do not correspond to distances in an n-dimensional hyperspace and the triangle relation holds only approximately, so some clusters of cases may be searched unnecessarily and some good matching cases may be missed, but these disadvantages are outweighed by the increase in speed of search (as the tests have shown).

If a good matching case is found in a leaf then the search is stopped, otherwise, other branches of the tree are searched (backtracking). In order not to search the whole tree, when some atypical image is being classified, the total number of the leaves searched is limited to a predefined number of leaves with the same class as the first leaf found and a predefined number of the remaining leaves.

The difference between the clustered and the flat organisation is clearly visible when searching for a case that already is in the case-base. In the flat organisation, on average, half of the cases in a leaf have to be matched. In the clustered organisation usually all the matches with the representatives R have to be calculated, but after sorting on D_R has been done the matching case is found in the first few clusters.

The retrieval algorithm can return an arbitrary number of best matching cases (currently three) each of which can subsequently be evaluated. However, the test have shown that this results only in a minimal gain in the total recognition percentage — about 0.5%.

Retrieval tasks done in parallel. If the operating system on which the software runs supports multithreading it is relatively easy to parallelise the search algorithm by using several search threads. When a multithreaded program is then run on a multi-CPU hardware a speed gain will be achieved. The lower the interdependency between the threads, the higher the speed gain will be.

Figure 7.20 illustrates the parallel search algorithm as implemented for searching a collection of clusters of cases. Each thread processes its own cluster and when finished moves to the next cluster that has not been processed yet. There is little interdependency between the threads except for common indexes to the best matches (a local copy of the case being matched is kept in each thread). Parallel processing in separate threads is also used for simultaneous searching of the two case-bases (see Figure 7.15).

Though, the parallel searching algorithm has already been implemented, it has not been extensively tested yet. However, based on, for example, the results reported in [Post & Siering, 1998], speed gains of about 75% seem realisable by running the software on a dual, instead of single, Pentium II hardware.

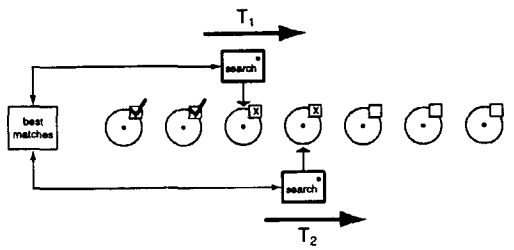


Figure 7.20: Searching clusters of cases using multiple search threads.

7.5.7 Match evaluation

After a search is finished, its results, i.e., the retrieved cases and the match values, are evaluated by a set of rules using a CLIPS expert-system shell, see Figure 7.15. The information about the search results is passed to CLIPS as the following set of facts:

- *For the retrieved non-defect image:*
  - current image data *(A)* – numbers of subclusters, channels, elements and incidents present, but no element details,
  - retrieved image data *(B)* – same type facts as for A,
  - data about the elements of the current image that match with the elements in the retrieved image $(A \cap B)$,
  - data about the elements of the retrieved image that match with the elements in the current image $(B \cap A)$,
  - data about the elements of the current image that do not match with the elements in the retrieved image $(A-(A \cap B))$,
  - data about the elements of the retrieved image that do not match with the elements in the current image $(B-(B \cap A))$,
  - location match for the subclusters,
  - individual match values for all the subclusters – match between A and B,
  - individual match values for all the subclusters if non-matching elements are removed, i.e., a match between $A \cap B$ and $B \cap A$,
  - global A and B match returned by the search algorithm,
  - and the classification belonging to the retrieved image.
- *For the retrieved defect image:*
  - global match returned by the search algorithm,
  - the classification belonging to the retrieved image.

These facts are processed by a set of rules. In the prototype version only a few rules are used. These can be divided into three groups:

- *Rules operating on global-match values only* — These rules look only at the values of the global matches with the retrieved defect and non-defect images and decide about acceptance, rejection, match uncertainty, etc., The following rules are present (in the order of priority):
  - A rule detecting possible inconsistencies manifested by high values of matches with both defect and non-defect cases. The non-defect case will be flagged as unreliable, and not used until later inspected. The identifiers of both cases and of the current image will be written to a log file.
  - A rule detecting high matches with defect cases. The image will be shown to the operator even if the non-defect-case match is even higher.
  - A rule detecting whether the match with a defect case is higher than the match with a non-defect case. For the sake of reliability the image is shown to the operator.
  - A rule detecting a very low non-defect match. No conclusions can be drawn, the image is shown to the operator.
  - A rule detecting high non-defect match. The image is automatically classified.
  - A rule detecting whether the match with a defect is close to the match with a non-defect and the classification might be unreliable.
- *Utility rules* — These do processing of the initial facts, extracting information and storing it as facts which are easier for the other rules to process. They do the following:
  - collecting information about the subclusters that were matched – non-matching elements, number of extra incidents, etc.,
  - finding the extra subcluster (if present) and determining the channels present in it.

- *Advanced rules* — These are rules relaxing some of the match conditions, where possible, to assign classifications even if the global match was below the automatic-classification threshold. The following rules are present (listed roughly in the order of importance):
  - A rule lowering match requirements for noise images (that is for situations where the retrieved image has only noise (STO) classification). A lower location match is allowed plus the current image is allowed to have non-matching single-incident elements and the retrieved image is allowed to have non-matching two-incident elements.
  - A rule lowering match requirements for an image having a high match with a PSK-object image, the only difference between the two being extra bottom-echo-loss (BEV0) incidents.
  - A rule classifying images with a good "same" ($A \cap B$ and $B \cap A$) element match and where the difference in the number of elements is limited to the non-defect retrieved image having small extra BEV0, +70° and -70° elements. It is assumed that any defects would be manifested as extra image elements and not as missing elements.
  - A rule similar as above but applicable only if the retrieved case is a fish-plated joint. It allows for presence of additional BEV0 elements in the current image.
  - A rule similar as above but it allows also for BEV-only subclusters to have match as low as 0.8.

From the above rules, the first group, evaluating the global-match values is necessary in the system, but these rules could have also been hard-coded due to their simplicity. The "advanced rules" are significantly more complex, and they require use of an expert-system shell. Altogether the "advanced rules" increase the amount of images automatically recognised by the system by about 10%, which corresponds to about 2% increase in the total recognition ratio, as the tests have shown.

7.5.8 Interaction with the user

If an image has not been automatically classified by the match evaluation rules it has to be shown to the operator who will have to give it the final classification. In the current URS^{nt} system the classification is optional and as a result in practice only question-point (V) and defect (F) classifications are given. The CBR system, however, requires all images to be classified in order to operate properly.

Because the goal of the system is to reduce the workload on the operator, the necessity to enter the classifications should not negatively influence the speed of work (a row of function keys can be used to enter the classifications as was done during evaluation experiments). When a sufficiently good match in the case-base has been found then the classification for the found image can be used as the default classification for the current image. (It has been observed that when the case-base contains more than 5000 cases then, for all not automatically classified clusters, the suggested classification is in 72% cases correct, and in 21% cases only the suggested type of noise is incorrect, for instance STO_OTH instead of STO_W70). In case of worse matches it is better not to use their classifications as the suggested defaults, because wrong suggestions could reduce confidence in the system. If a good match with a defect case has been found then the operator should be warned about it. Thus, three general situations can be distinguished (apart from the automatic classification):

- The system suggests that the image has no defect and it has a certain classification.
- The system warns that the image is similar to an image with a defect.
- The system informs the user that it cannot recognise the image at all.

Certainly, during the initial phase of use it would be advisable if the operator could also see the most similar image(s) retrieved from the case-base(s) to check the suggested computer classifications. In case of doubts it should be possible to print out the images to have them checked later.

7.5.9 Case-base maintenance

In the prototype no attention has been paid to the maintenance of the case-base as it is relatively unimportant for the rather small amounts of data processed during the experiments. However, case-base maintenance is important for good working of the system over long periods of time. The two most important maintenance tasks are:

- *Consistency checking* — It is important that the classifications of the cases in the case-base are correct, otherwise either wrong classifications will be made or the performance of the system will deteriorate.
- *Case-base pruning* — Many of the cases entered into the case-base will be unique, that is not similar to any other. In later use they may never be retrieved, however, they do take up space in the case-base and a match with them may often have to be calculated, reducing this way the overall search speed. Such cases can be easily identified and should be removed. Preliminary tests have shown that removal of as much as 25% cases from the case-base lowers the recognition ratio only about 0.1%, while giving a significant increase in speed. (In the tests described in the next section no use of pruning was made.)

7.6 CBR system evaluation: experiments and results

7.6.1 Test data set

All the data ever acquired with the ultrasonic rail-inspection system is available on optical disks. It would seem that it would be easy to obtain a test set, however, it turns out that only the computer-classified images have the detailed classification (e.g., 0° noise, bolt hole), all the remaining images were given a classification only when a possible defect was present. This means that the available data had practically only a defect/non-defect classification. Several test runs were done on this data, however, the results did not provide the required detail of information about what type of images are recognised and what not.

Because of this it was necessary to prepare a set of fully classified files which could be used for testing. A classification scheme slightly different from the one used on the inspection train was used, which could assign three classifications to each image:

- OBJ classification — it tells what construction object is visible, if none is visible then SPS (rail) class is given,
- STO classification — it tells what kind of distortion/noise is present,
- VFP classification — it tells what kind of defect or question point is present.

Table 7.4: Classification codes used to prepare the test set.

OBJ — object	STO — noise	VFP — defect
SPS (rail)	NON (none)	NON (none)
BG (bolt hole)	BEV (bottom-echo loss)	ZVB (kidney-shaped crack)
B2 (double bolt hole)	S0 (0° noise)	OTH (other)
BGO (bolt-hole object)	W70 (water-interface disruption)	
T0 (thermit weld with bolt holes)	OTH (other)	
PL (fish-plated joint)		
PSK (frog)		
WSL (switch)		
OTH (other)		

This classification scheme makes sense because all three types can be present in one image, for example, there can be a fish-plated joint with some noise and a defect. Not all classes from Table 7.1 were used because some of them are very rare, and those used were enough to get an impression about the performance of the system. The classification codes that were used are listed in Table 7.4.

After a period of working on the project we had enough experience to be able to do classification without the help of the RGS operator, though it took more than a week to fully classify a set of 128 normal data files and 12 files containing defect images (extracted from a large number of normal data files). The test data sequence consisted of 6 randomly ordered defect files (to fill the defect case-base), 128 randomly ordered normal data files, and again 6 defect files (to test the system reliability). Altogether, the test set contained approximately 38000 non-bottom-echo-loss images from approximately 1000 km of track (this corresponds to estimated 4 weeks of inspections with the old train). Figure 7.9 shows a summary of the contents of the non-defect files.

7.6.2 Test environment and software

Because the processing speed is an important factor in evaluating the system it is important to note here that all the tests described in this section have been done on a single CPU-300MHz Pentium II computer with 64MB RAM. Unless otherwise noted, the tests were performed before the C4.5 decision tree was used in the case-base organisation, however, the difference between the two case-base organisation types has a minimal influence on the results (see Section 7.6.4). Because the data files correspond to a certain length of inspected track, the system performance is expressed in kilometres of track-data processed per hour (km/h).

7.6.3 Rule-based versus case-based versus hybrid classifier

At the design stage a decision was made to make use of both a rule-based and a case-based classifier. To verify the correctness of that decision the test scan was done three times: once with the CBR part turned off, once with the rule-based part turned off, and once with the full functionality. This section contains a short comparison of the results obtained with the three methods, Section 7.6.5 contains the full analysis of the results for the hybrid system. The comparison of the overall classification results is shown in Figure 7.21. The rule-based classifier has the lowest recognition ratio,

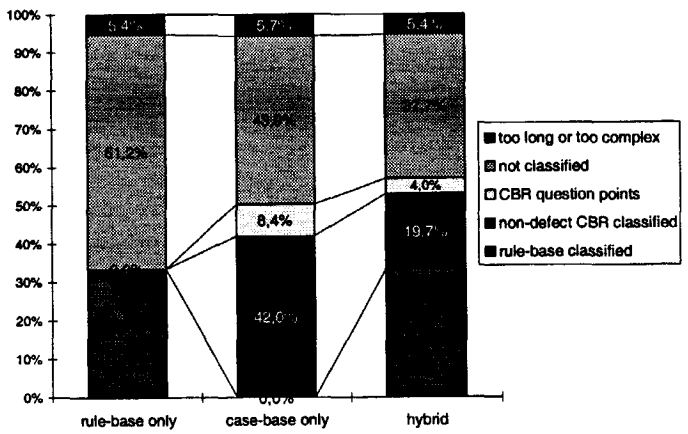


Figure 7.21: Comparison of classification results for rule-based, case-based, and hybrid classifiers.

however, it was approximately 14 times faster than the other two classifiers (the average processing speeds were: 401 km/h for the rule-based classifier, 23 km/h for the case-based classifier, and 28 km/h for the hybrid classifier). Important things that can be noticed are:

- The rule-based classifier is capable of recognising some complex noise images which are unrecognised by the case-based classifier (5.7% versus 5.4%).
- CBR question points are those images which had a close match to a defect case, the rule base recognises some of them correctly as noise, while the case-based classifier shows them to the operator (8.4% versus 4.0%).
- In the hybrid system the case-based classifier uses information about subcluster classifications (from the rule-based classification) during matching and match evaluation, this information is not available in the case-based-only system.

It is clear that the results of the hybrid classifier, as far as the recognition ratio is concerned, are the best. It has certainly advantages over the case-based-only classifier especially as the rule-based part is already being used and can easily be incorporated into the new system. In spite of a much lower processing speed than that of the rule-based classifier, the adaptability and maintainability of the CBR system is considered as an advantage big enough to outweigh the disadvantage of longer processing times.

7.6.4 Comparing various ways of case-base organisation

The tests done with the C4.5 tree organisation gave slightly (a fraction of a percent) worse recognition results compared with the old fixed 3-level tree organisation probably because a smaller portion of cases was searched due to the limit on the number of leaves searched. The speed was minimally slower than before. It seems that the old organisation was already quite optimal, moreover, it was simpler thus its implementation was faster. The C4.5 tree organisation may also suffer from the fact that a sizeable fraction of the images belonging to the same class are significantly different from one another (the old organisation was less class-oriented and more oriented towards the image similarities as defined by the matching algorithm). Also, the test has shown that almost a half of all the cases stored in the case-base are stored in one leaf corresponding to the fish-plated-joint class. Still, it has been decided to retain the decision-tree organisation as it offers a more flexible way of case-base organisation (it is possible to define the old case-base structure by editing the configuration file by hand), and possibly with some further code or tree optimisation it will also bring speed gains.

Another test was done to compare the clustered leaf organisation to the flat one. The test was done on our full data set of 140 files (under slightly different conditions then the test for which the results are shown in Figure 7.21). The flat organisation resulted in a total processing time of 46½ hours while using the clustered organisation the processing took 38¼ hours to complete, which gives an overall speed gain of about 18%. This gain should be more pronounced as the case-base gets larger, as a test done on 70 files gave only an 11% speed gain. Because searching in the clustered case-base is not exhaustive there was theoretically a possibility that the recognition ratio would fall. It did fall, but slightly, but only by 0.3%, which corresponds to only about 100 images from the total of 36000 processed.

7.6.5 Results for the hybrid system

The scan of the 6 "defect" files resulted in two case-bases: a non-defect case-base with 120 cases, and a defect case-base with 803 cases. Afterwards, the normal "non-defect" data files were scanned, followed by the remaining 6 "defect" files. The individual files contained from 25 to 2039 clusters. Presence of big and small files may skew the results a bit but a running average over several files is

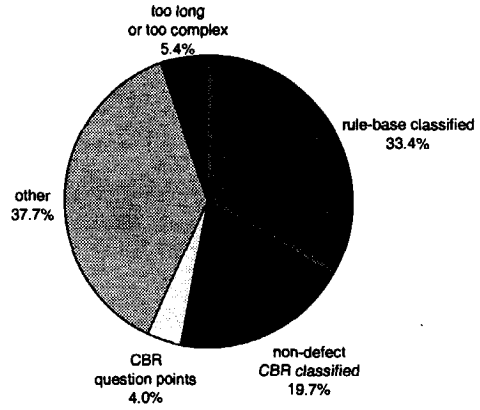


Figure 7.22: Overall results for the hybrid system.

used in the graphics with the results anyway. At the end of the scan the non-defect case-base contained approximately 13000 cases (15.5 MB) and the defect case-base contained 1700 cases (1.7 MB).

The overall recognition ratio for the hybrid system is shown in Figure 7.22. The recognition percentages for the various categories of images are shown in Figure 7.23. It is clear that the system is good at recognising bolt-holes (single or multiple) and S0 noise. Worse results are achieved for images of fish-plated joints. The PL images recognised by CBR are usually the "ideal" ones, more complex images usually give a low match. Only 27% recognised fatigue cracks does not mean that some of these defects were missed by the system, the remaining 73% were classified as belonging to the question-point class and would be shown to the operator.

For a CBR system, an increasing recognition ratio with an increasing size of the case-base can be expected. This trend is visible especially at the beginning of the scan where the CBR system quickly (after approximately 30 data files) learns to recognise simpler images like bolt holes and 0°

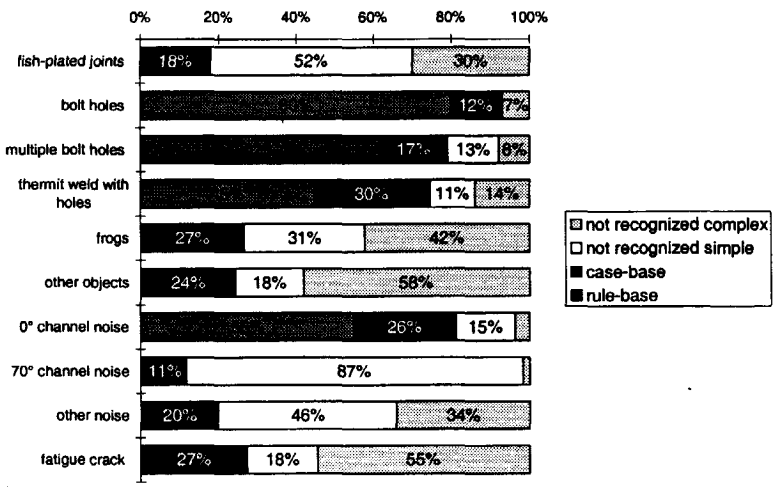


Figure 7.23: Average recognition ratios for various image types for the hybrid system. Possibility for improvement lies mainly in the 'not recognized simple' area.

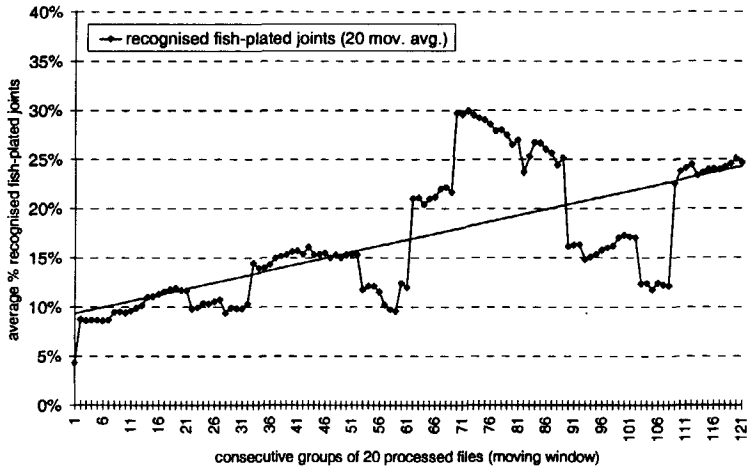


Figure 7.24: Increasing fish-plated-joint recognition ratio with an increasing size of the case-base (percentages are calculated per 20 processed data files (moving average)).

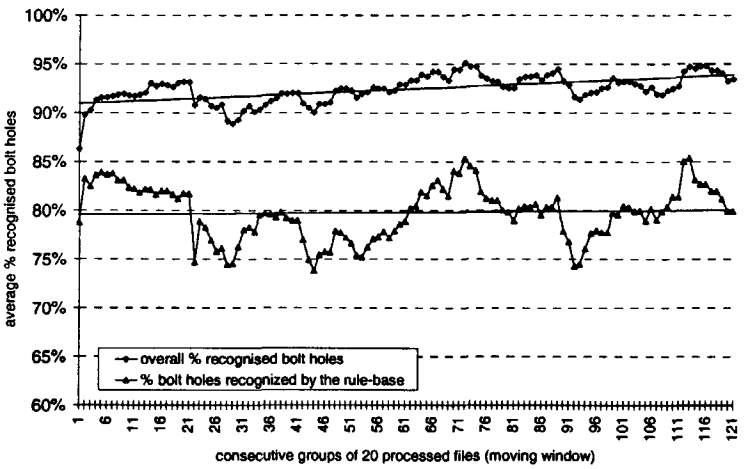


Figure 7.25: Bolt-hole recognition percentages.

noise. It takes much longer to improve the recognition ratio of the more complex images like fish-plated joints (see Figure 7.24). Figure 7.25 shows a "cooperation" of the rule-based classifier and the CBR in recognising bolt holes. The recognition ratio of the rule base is on average constant. The CBR system recognises $^1/_3$ to $^1/_2$ of the remaining bolt holes. It is also visible that the recognition ratio of the CBR system rises slowly with the increasing size of the case-base.

The processing speed expressed in km/h is shown in Figure 7.26. The speed with a case-base of 13000 cases falls to around 25 km/h. It can also be noticed that the fall in processing speed is slower as the case-base becomes larger; this was to be expected because of the way the search algorithms are designed (using the clustered case-base organisation and breaking-off of matching). The average processing speed per file varies widely independent of the size of the case base. It depends on the

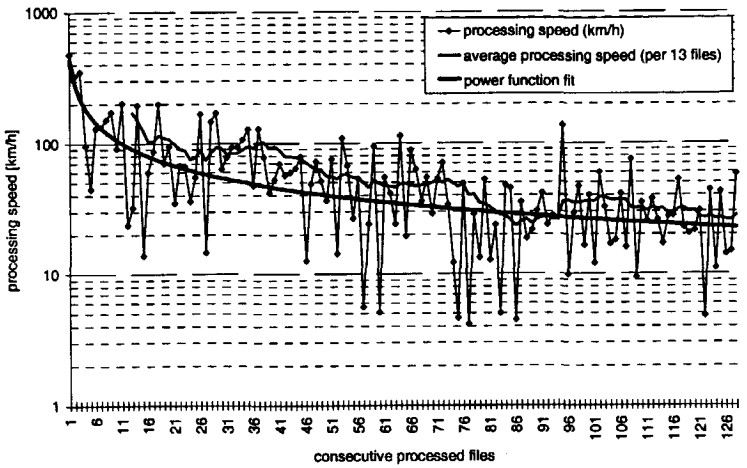


Figure 7.26: Processing speed on a 300MHz Pentium-II PC with 64MB RAM (beginning with an empty case-base).

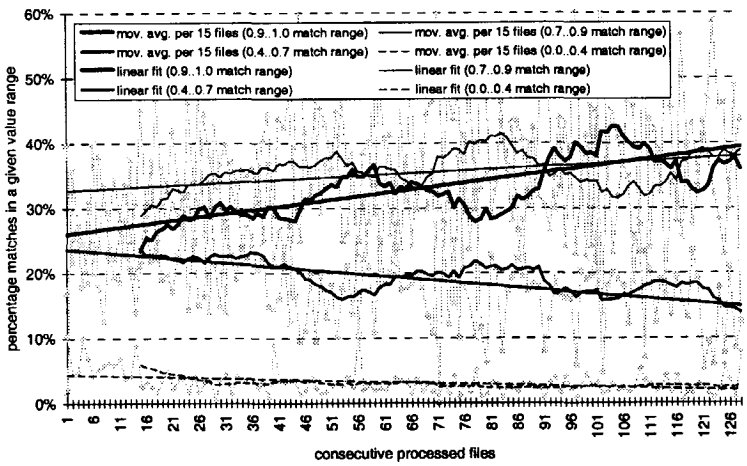


Figure 7.27: Values of the case-base matches.

number of image clusters per km track, but also on the type of images. Especially fish-plated joints require long processing time because they are complex and because there are so many various fish-plated joints in the case-base.

An interesting parameter to analyse are the values of the best matches with cases in the case-base, see Figure 7.27. They give an indication how good the matches with the cases are, even for the matches below the automatic-recognition threshold. Increasing numbers of matches just below the threshold indicate that the system will further improve its performance as the number of the cases in the case-base increases.

7.6.6 Disagreements in classification

During the tests all the disagreements between the (pre-)classifications present in the files and the classifications given by the systems were stored to report files. The reported disagreements were later analysed. In 73% of classifications there were no disagreements. For 8% of cases a differing noise classification was given, which is a minor mistake of no consequences. Another insignificant difference is in a different question-point classification; there were 2,6% of such. In 12% of cases a noise classification was confused with question-point (or vice-versa). This is mainly due to ambiguous images which were inconsequently pre-classified. This leaves about 4.4% other, possibly more serious misclassifications, which were analysed in detail.

Misclassified defects and question points. The more important disagreements are those when a VFP_OTH or VFP_ZVB cluster is automatically classified as VFP_NON. During the scan 1647 of VFP_OTH or VFP_ZVB clusters were scanned in "defect" files and 322 in "non-defect" files. Of all these clusters 12 were given VFP_NON classification. These 12 disagreements can be divided into four (overlapping) categories:

- mistakes in pre-classification — 4 misclassifications,
- ambiguities — 8 misclassifications,
- wrong or uncertain rules — 2 misclassifications,
- possible imperfections in the clustering and matching algorithm — 2 misclassifications.

The two last groups of misclassifications were corrected after the test by making small changes to the rules and to the parameters of the matching algorithm.

Other misclassifications. Other less serious disagreements in classification can be divided into 2 groups:

- wrong or inconsequent pre-classifications — meaning that the cases in the case-base had (possibly) incorrect classifications.
- ambiguous situations where more than one classification is possible.

7.6.7 Discussion of results

An observation that can be made based on the scan results is that there is much variation in data between the files. It was attempted to counter it by scanning the files in a randomly chosen sequence and using running averages to analyse the data. Still, it is difficult to determine trends in the results.

The hybrid rule-based/CBR system achieves better results than the rule-based only system. The system achieves higher than 50% recognition ratio, and should improve with the increasing case-base size. For some files recognition ratio of more than 70% could be reached. The theoretically highest achievable recognition ratio is approximately 95% because of 5% images which are too complex to be processed by the image segmentation algorithm.

Figure 7.23 shows that the system is good at recognising BG, BG2, BGO, and S0 images. The recognition ratios reach 80 to 100% using a combination of rule-based and CBR classification. The image category which presents greatest problems are PL images which form a large percentage (about $^1/_3$) of all non-BEV images. There is a large variation between PL images and that is why few can be recognised by the system; however, it has been observed that while scanning the last files a high proportion of PL matches had value 0.6 or higher. This suggests that with growing case-base size more PL images will be recognised.

Lowering the acceptance threshold would lead to a higher recognition ratio, however, the influence on the reliability of the system would have to be investigated. Adding more match evaluation rules is a more reliable way to increase the recognition ratio; however, it increases the complexity of the system. The three specific PL rules (see Section 7.5.7) were responsible for recognition of 348 out of 2084 CBR-classified PL images.

The case-based classification has one main disadvantage: the long processing time. If it is assumes that on average 1500 non-BEV images have to be processed per day, then the highest average recorded processing time per cluster of approximately 11 sec (avg. was about 4 sec) would result in roughly 4.5 hour processing time per "day" of data. This value should be reduced by a factor of at least 3 (or even more if a larger case-base is to be used) to give a useable system. Such a speed-up is achievable by improving the algorithms used and their implementation, and a use of faster hardware, see Section 7.8.2.

7.7 Conclusions

The problem of B-scan image classification in the URS cannot be solved solely using rule-base classification. There is a large variety of images and the available knowledge about them is not well formalised, namely, the fish-plated-joint images are a problem. Still, a relatively small set of rules can classify almost one third of the non-BEV images. The CBR system could classify further 20% of images increasing the recognition ratio to 53%. This recognition ratio should improve further with the increase of the number of cases in the case-base. Improvements are still possible, like: better noise removal, adding more evaluation rules, fine tuning of the matching algorithm. A disadvantage of the CBR system is the long processing time. This time can be shortened by improving the software, especially the case-base organisation, but if the system will be used in practice it will require fast hardware.

Apart from doing the classification, the CBR system makes it possible to gather information about the types of images present in the data files and other statistical data. This way for instance good candidate images can be identified to be described by rules (e.g., some S0 images).

Though all the data ever acquired by the system was available on optical disks, still it could not be directly used as cases for the CBR system because the data lacked full classification. A considerable effort was required to classify a number of files to obtain a test set. Though the system can theoretically be taken into use with an empty case-base, still a set of cases is needed to do the necessary test. This indicates a general problem in building CBR systems, namely, that availability of data does not necessarily mean availability of cases.

7.8 Future work

7.8.1 Deployment on the train

Currently (fourth quarter of 1998), work is being done on implementing the system for the inspection train. First, a simpler batch version of the system will be implemented, as illustrated in Figure 7.28b. In such a configuration, data from a stretch of rail track is first processed by a rule-based system, then by the CBR system, and then the remaining not classified images are classified by the operator. Each of the systems processes a full batch of data and writes the results in a classification file. A disadvantage of this configuration is that the system cannot learn directly from the classifications made by the operator — these will be processed once per day. An advantage is that because of the use of existing, almost unchanged, modules for the rule-based and interactive classification the cost of implementation is lower and the changes in the current way of working will be minimal. Also, the speed of a batch system is less critical than the speed of a fully integrated interactive system, like the one in Figure 7.28c. The responsiveness of a fully integrated system can be improved if its implementation makes use of pipelined parallel (instead of sequential) processing, where the operator could take over from the automatic system if it was lagging behind.

Because, in real use, more cases will be processed than during testing, provisions will have to be made to keep the size of the case-base manageable, for example, by limiting case-base to some maximum size as suggested in [Surma & Tyburcy, 1998]. Moreover, a case-base maintenance system

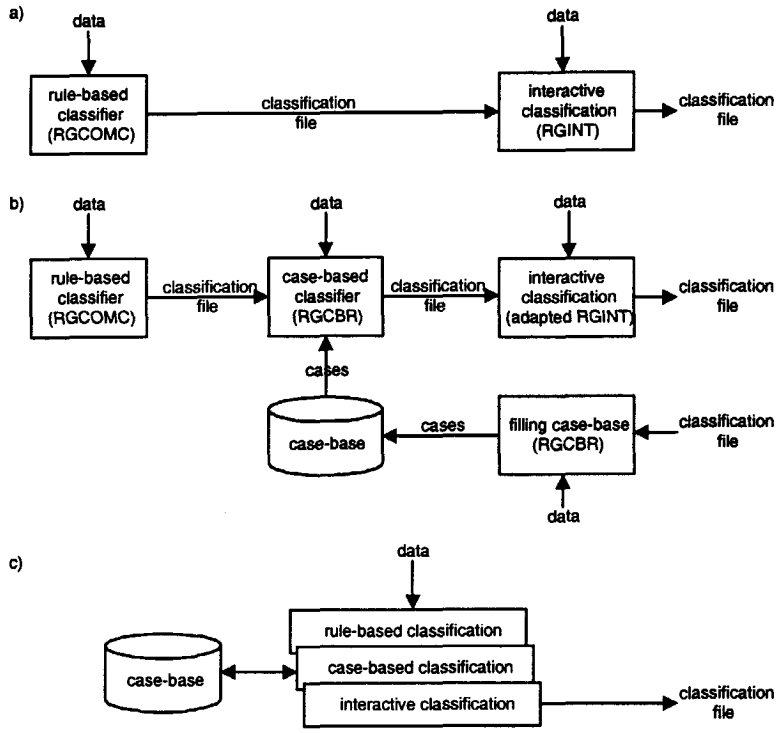


Figure 7.28: Phased deployment: a) current system, b) system extended with CBR capabilities (batch mode), c) goal system.

will have to be developed. It should provide means for consistency checking, correction of mistakes, and operator assisted pruning.

The three-code (OBJ, STO,VFP) classification scheme, which was useful for the experiments with the prototype (in particular detection of misclassifications) will be changed to a two-code scheme (an object code plus a combined noise/defect/question-point code) which closely corresponds to the scheme currently implemented in the interactive classification software (RGINT).

The system should not be difficult to convert to the new 8 channel acquisition system used in URS^{nt}. For this sake the rule base of the rule-base classifier will have to be adapted, the clustering and matching algorithms extended, and the evaluation rules in the CBR system adapted and added. These are not significant changes, as they will mostly copy the existing design.

To achieve required processing speed the system will be implemented on a dual-CPU hardware. Initial estimates from available benchmarks [Post & Siering, 1998; Stiller, 1998] are that a 2-CPU Pentium 400MHz system should be sufficient to achieve at least two-fold increase in speed as compared to the system used in the tests.

7.8.2 System improvements

Availability of faster hardware should make it easier to fine tune various parameters (like: optimising the d-tree and the clustered case-base organisation, the matching algorithm, and possibly adding new rules) by reducing time necessary to run the tests. Improvements of the inspection system hardware, like an addition of a fish-plate detector, can also contribute to the increase of the recognition ratio of the system.

As the noise in the images is a significant problem in the classification, we are going to look for better ways of noise removal. Noise can be a problem in CBR because it undermines the assumptions about the regularity of the world. The solution is not to let the noise become part of the problem description or to properly identify it as noise (and reduce its importance in the match), which means that the noise has to be recognised in the situation assessment stage of the CBR.

In the future, adding some context information to the cases could be considered, such as adding the classifications of the neighbouring (on the rails) images and the so-called markers recorded during the ride (e.g., for level crossings). Use of this information could lead to improved reliability and/or increased recognition ratio.

The system could also be adapted to record failed suggestions made on the basis of cases. These can be used to detect ambiguous (or incorrectly classified) cases. No facilities to acquire explanation for these failures will be added because these would require extra interaction from the operator. All extra interaction is perceived as an extra work that has to be done, and might require changes to the system interface and the way of working with it, which is undesired.

7.9 References

Bray, D. (1991) "Ultrasonic Applications in the Railroad Industry", *Nondestructive Testing Handbook, Vol. 7, Ultrasonic Testing*, A.S. Birks, R.E. Green, Jr. and P. McIntire (eds.), ASNT, pp. 594–634.

Esveld, C. (1989) *Modern Railway Track*, MRT-Productions, Duisburg, Germany, pp. 263–276.

Giling, E.J.M. (1996) *Detailontwerp Software Werkstation URS^{nl} - Versie 1.0*, TNO report, TPD-FSP-RPT-960058.

Gonzalez, R.C. and Woods, R.E. (1993) *Digital Image Processing*, Addison-Wesley, Reading, MA.

Havira, R.M. and Chen, J. (1995) "High Speed Rail Flaw Pattern Recognition and Classification", *Proc. SPIE*, Vol. 2458, pp. 64–73.

Hopgood, F.F., Woodcock, N., Hallam, N.J., and Picton, P.D. (1993) "Interpreting ultrasonic images using rules, algorithms and neural networks", *European Journal of NDT*, Vol. 2, No. 4, April, pp. 135–149.

Jansen, E. (1991) *Kennissysteem voor classificatie van de meetgegevens van de Ultrasoontrein*, M.Sc. Thesis, Rijksuniversiteit Groningen, The Netherlands.

Jarmulak, J. (1995) *Second semi-annual report Andes project: Rule-base (URS) an CBR (EC) prototypes*, TNO-report, FSP-RPT-950090.

Jarmulak, J. (1996) "B-scan Image Clustering and Interpretation in Ultrasonic Rail-Inspection System", *Proceedings of the second annual conference of the Advanced School for Computing and Imaging, Lommel, Belgium, June 5-7*, pp. 190–195.

Jarmulak, J., van't Veen, P.P., and Kerckhoffs, E.J.H. (1997) "Case-Based Reasoning in an Ultra-sonic Rail-Inspection System", *Case-Based Reasoning Research and Development: Second International Conference on Case-Based Reasoning, ICCBR-97, Providence, RI, USA, Proceedings*, D.B. Leake and E. Plaza (eds.), Springer-Verlag, Berlin, pp. 43–52.

van Katwijk, R.T. (1996) *The application of an expert system for the inspection and maintenance of contact wires*, TNO-report, FSP-RPT-960070.

Kolodner, J. (1996) "Making the Implicit Explicit: Clarifying the Principles of Case-Based Reasoning", *Case-Based Reasoning: Experiences, Lessons & Future Directions*, D.B. Leake (ed.), AAAI Press, Menlo Park, CA, pp. 349–370.

Krishnapuram, R. and Freg, Ch.-P. (1992) "Fitting an unknown number of lines and planes to image data through compatible cluster merging", *Pattern Recognition*, Vol. 25, No. 4, pp. 385–400.

Lesiak, P. (1992) "System for Automatic Ultrasonic Quality Control of Railroad Rails", *Russian Journal of Nondestructive Testing*, Vol. 28, No. 7, pp. 383–388.

Post, U. and Siering, P. (1998) "Doppel-Hertz: Zwei Prozessoren im Desktop-PC", *c't - magazin für computer technik*, No. 12, pp. 196–202.

Quinlan, J.R. (1993) *C4.5: Programs for Machine Learning*, Morgan Kaufmann Publishers, Inc., San Mateo, CA.

Roos, J. (1990) "Het ultrasoon railinspectie systeem", *NAG Journal*, Nr. 105, november, pp. 21–34.

Roos, J. (1995) *Basisontwerp RGS*, TNO-report, FSP-RPT-950078.

Sladojevic, B. (1995) "Ultrasonic testing of rails", *INSIGHT*, Vol. 37, No. 12, December, pp. 987–991.

Stiller, A. (1998) "Edles Gemisch: Intels Xeon für Workstations und Server", *c't - magazin für computer technik*, No. 14, pp. 162–165.

Surma, J. and Tyburcy, J. (1998) "A Study on Competence-Preserving Case Replacing Strategies in Case-Based Reasoning", *Advances in Case-Based Reasoning: EWCBR-98 Proceedings*, B. Smyth and P. Cunningham (eds.), Springer-Verlag, pp. 233–238.

Surma, J. and Vanhoof, K. (1998) "An Empirical Study on Combining Instance-Based and Rule-Based Classifiers", *Proceedings of the AAAI'98 Spring Symposium on Multimodal Reasoning*, AAAI Press, pp. 130–134.

Tange, I.J. (1990) *Automatisch klassificeren van spoorstaafobjecten*, Afstudeeropdracht HTS Dordrecht, The Netherlands.

van der Werf, G. (1993) *Visuele inspectie voor beslissingsmodel bovenbouwvernieuwing*, M.Sc. Thesis, Delft University of Technology, The Netherlands.

Chapter 8

Discussion and Conclusions

8.1 Why and how to automate NDT data interpretation?

NDT costs money, thus organisations doing NDT must have valid reasons to spend their money on NDT. One reason can be that inspections simply have to be done because of the existing national and international laws and regulations. Another reason is that by doing an inspection one expects to save money eventually. It might seem that not doing the inspection would save even more money, but, as Chapter 2 has shown, this is a wrong approach. Cost of inspection can be considered as an investment, the return on which comes in the form of safe and failure-free, thus less expensive, operation of the inspected structures and products. In other words, one benefits more (in money-terms but not only) by doing the inspection than by not doing it. This may be not true in situations when the inspection is prescribed by external regulations (laws); though, one would expect that the regulations are made on a basis of careful analysis which balances the benefit (safety) and the cost.

One way to increase the added value (the benefit minus the cost) of an inspection is by automated interpretation of the NDT data. The main benefits of automated interpretation are: increase in the speed of inspection, increased reliability (increased POD and decreased PFI) and increased accuracy and consistency. Interpretation of NDT data is a complex task which requires use of AI techniques or related techniques from pattern recognition and/or statistical classification. Chapters 3, 4, and 5 have presented several such AI and non-AI techniques which can be used for data interpretation. Use of any of these techniques to build a system for NDT data interpretation makes sense only if the cost (often significant) of such a system can be earned back by increased added value of the inspection performed using the system. This may mean that a simple system interpreting 5% of the data but easy to build and needing no maintenance at all, may, from an economic point of view, be better than a sophisticated system in principle capable of replacing the operator, but costing huge amounts of money to build, even more money to maintain, and never gaining full trust of the users. So, in other words, the question is not "Can it be built?" but "Does it make sense to build it?".

8.2 What is so special about interpretation of NDT data?

In this thesis, interpretation NDT data has been discussed. The obvious question is if the fact that we are considering interpretation of NDT data makes that much difference in the choice of the most suitable data-interpretation technique. Would not just any data-interpretation technique be sufficient?

There seems to be a number of factors distinguishing the NDT data interpretation problems which significantly influence the choice of data-interpretation technique. The three which are most important are (see also Section 5.2.1):

- *Type of data* — Data from NDT systems is in the form of signals, images, etc. These are usually difficult to interpret. "Traditionally" this type of data is processed by statistical or ANN classifiers (e.g., character recognition); however, these may be unsuitable for specific NDT problems (see next point). The use of expert systems or CBR for this type of data has not been common.
- *Reliability* — NDT data-interpretation systems have to be reliable. Nobody can be expected to double-check the results of the automatic interpretation. Moreover, mistakes made by the system may have serious consequences. This distinguishes NDT from many other application

fields of AI, where, either the cost of mistakes is negligible or the results are anyway presented to a human who can always check them if necessary. The reliability is also an important factor with a view to the many laws and regulations in the NDT field.

- *Cost aspect* — Maybe except for nuclear-energy and aeroplane industries money is never in overabundance in NDT. This means that only relatively inexpensive data-interpretation techniques have a chance of success in NDT. Because the cost of a data-interpretation system lies largely in the knowledge acquisition stage and in the future maintenance, the techniques for which these costs are low are especially attractive for NDT.

8.3 Which technique is best?

8.3.1 The requirements

When choosing a technique to solve a data-interpretation problem, the first selection step is of course the question: can the technique do what it is required to do? A technique has to be chosen which can interpret the data coming from the inspection system, as some techniques may be unsuitable for some data-interpretation problems. For instance, ANNs may be unsuitable for classification of certain forms of data, for example, the B-scan data from the URS system described in Chapter 7, would be difficult to represent in a form suitable for an ANN classifier. A related problem posing difficulty to some techniques is the existence of much variety in the data to be interpreted. If this is the case then, for instance, obtaining a good data set for training a classifier can be practically impossible.

To reduce the construction costs, it may make sense to lower the requirements. It is well known that often making a system recognise the last 20% of the data may constitute 80% of all the construction cost. Therefore, setting the requirements lower may result in an overall better system (in the light of benefit/cost discussion in Section 2.3.3). Still, one has to choose a technique which can identify the data it cannot recognise, which brings back the always present requirement for the high reliability of the NDT data interpretation systems. The required high reliability may be a determining factor in the choice of the data-interpretation technique, because some techniques are inherently less reliable than others. The characteristics which can lead to a lack of reliability are, for example: too wide generalisation – present in some classifiers; and the closed-world assumption – present both in some classifiers and in expert systems.

Once it is known which techniques are capable of doing the required task, one has to choose the technique which will cost the least. Two main cost factors can be distinguished: cost of construction (knowledge acquisition and building the system) and cost of use (largely determined by the ease of use and the cost of maintenance).

8.3.2 Cost of construction

It is difficult to say which of the two, the knowledge acquisition or actually building the system, is more expensive. It depends much on the type of problem, though, generally, building cost can be low when learning techniques and/or standard software tools are used.

Knowledge about data interpretation is available mainly in two forms: as the already classified data and as the expertise of the NDT inspector. The problem with the data is that it is often not complete: there may be too few defects in the data set, or too little data from non-defects, or not all defect types are equally well represented. Use of a non-complete data set as a basis for building a classification system may result in low reliability of that system. Obtaining a good training data set may be quite expensive, requiring making special calibration pieces or doing a destructive analysis of real defects.

The "classical" knowledge acquisition, i.e., formalising the knowledge of an expert, may be difficult for NDT problems. The main difficulty is that the inspectors usually recall the standard cases (the typical defects), while it is precisely how the system can handle the unusual data which determines the reliability of the system. These exceptions are difficult to recall by the experts. If one is not careful, the exceptions may be discovered only when the already built system starts to make mistakes. Knowledge acquisition costs money, but also the time of the inspector, which may be a problem when building a data-interpretation system for a small inspection company.

While it is difficult to collect a complete data set and to acquire all necessary knowledge from an expert, it is, generally, easy to gather a small set of example defects and to acquire general knowledge about the problem domain. An ideal, from the viewpoint of construction effort, technique should be able to make the best use of both (without compromising reliability). Also, learning techniques can in principle contribute to keeping the development cost low.

It may be advantageous to build a hybrid system, combining several techniques, instead of a system based on only a single technique. By combining several simpler modules, use can be made of the specific strengths of each technique. These modules may additionally be simpler to develop, and easier for the users to understand and to maintain, as compared to a single monolithic system.

8.3.3 Cost of use

The cost of use is determined by two main factors. The first is dependent on the ease of use of the system. Compared to a non-automated system the new system may require use of automated scanning, careful calibration, and have a more complex user interface. All these may mean that the system will be more difficult to handle than the previously used system. This may increase the cost of inspection.

The second component of the cost of use is system maintenance. This may include making small changes necessary to adapt the system to the specific conditions of a given inspection. In case of learning systems it may be necessary to inspect the acquired knowledge for consistency and usefulness for future inspections. The most difficult cases of maintenance are those when a system turns out to be unsuitable for a new inspection, for example, because it cannot handle new data types, and the system has to be adapted by its developer.

8.4 How does CBR fare?

8.4.1 Advantages and disadvantages

An important advantage of CBR, when considered for NDT applications, is its reliability. The reliability is "built into" a CBR system at two levels. First, there is the inherent localised problem domain representation in the form of cases instead of wide generalisations used by many classifier types. Then, there are the design decisions where one has to find the means for the system to distinguish cases. The tests that have been done with the prototype URS-CBR and LISSA systems (Chapters 6 & 7) have indeed shown them to be reliable, the misclassifications which occurred were mainly due to ambiguities and could also have been made by a human operator.

CBR also makes use of a combination of knowledge representations (knowledge containers, see Section 4.3) which makes it easy to use both the available data and general expert knowledge to build the system. The use of readily available knowledge may contribute to lower development costs. Moreover, knowledge distribution means also knowledge structuring, which may help to build, to understand, and to maintain CBR systems. Indeed, designing both URS-CBR and LISSA use has been made mainly of general knowledge available either from NDT handbooks or coming from the operators. Because all the specific classification problems are handled by cases obtained from the data, the knowledge acquisition was relatively easy.

CBR systems are incremental learning systems, therefore, they will improve their performance during use, but only if they have somebody to learn from. Because of this, they are best applied where they interpret the data in cooperation with a human operator, fortunately, this is a usual situation in the NDT inspection.

CBR systems should be better than ANNs or expert systems in handling data in form of images and signals. An appropriate similarity measure can handle a wider variety of data than feature-vector-based data representations as commonly used for ANNs. CBR provides also more possibilities for representation of complex data than it is done by expert systems, especially the rule-based ones. The experience with the rule-based part of the hybrid URS system has shown that only relatively simple images could be efficiently described and handled using rules.

ANNs (or statistical classifiers) have an advantage over CBR when applied to simple interpretation problems (e.g., for homogeneous data) where good training sets are available. In such situations it is generally less expensive to use ANNs. When complex inhomogeneous data has to be classified or the training set data is scarce, use of ANNs may result in unreliable and inflexible systems. In such situations CBR is a better solution, because available knowledge about the problem domain can be used to compensate for the missing training data.

Rule-based systems are best suitable for well-understood and stable problems, where the knowledge acquisition is easy and there is little need for the knowledge maintenance. However, even in such situations CBR systems are a viable alternative, especially as the use of cases may reduce the complexity of the rules used and simplify maintenance. Especially, when the knowledge about the problem domain is weak and/or difficult to formalise then CBR systems have a clear advantage.

8.4.2 Issues when developing CBR systems

When developing CBR systems for eddy-current and ultrasonic B-scan data interpretation, three common issues have emerged as requiring the most effort in the system development: the case representation, case comparison (similarity measure), and collecting the data for system testing. Additionally, in the case of the URS system an efficient case-base organisation was a significant issue in the development of the system.

The case representation and the similarity measure are two related issues, and they are fundamental for the retrieval of the most relevant cases. They can be expected to be the most important issues in the development of CBR systems for NDT data interpretation. Other issues such as, for instance, adaptation and learning from user explanations are less important because in the classification problems rarely any adaptation is done and, usually, the user interaction will be limited to the confirmation or correction of the system suggestions. The problem of case representation in particular and of data interpretation in general can often be simplified by appropriate preprocessing of the data. Attention should be paid at least to noise removal.

For the practical application of CBR for URS data interpretation much attention had to be paid to the speed of the data retrieval. Speed became an issue because of the complexity of case matching and the large number of cases in the case-base. Retrieval speed has been increased by using a efficient case-base organisation. The slow retrieval speed and the large memory requirements for storing the cases are at present probably the largest disadvantages of CBR when applied to NDT problems. However, this should cease to be a problem within the next couple of years as the computers get faster and the prices of the memory get lower (see Appendix A). Because of these developments in computer hardware, CBR should become even more attractive in the future.

Apart from the advantages coming from faster and less expensive hardware, CBR should also profit from the more and more common digital data storage in NDT equipment and the use of software tools for reporting the results of inspections. This should simplify obtaining cases for system testing, as well as make it easier to add CBR functionality to existing computerised NDT inspection systems.

Development of CBR systems for NDT would be easier if suitable CBR development tools were available. However, few of the existing CBR tools provide sufficient facilities for representation and matching of the complex NDT data.

8.5 Problems in automating NDT data interpretation

Automated NDT data interpretation systems of the kind described in this thesis are still quite rare in the NDT world. Little is also heard about any new developments in this field. This might indicate that introducing automated data interpretation to NDT is not an easy task. The problem lies in the fact that analysing the impact of the introduction of new techniques is difficult because a large number of factors are involved as Chapter 2 has indicated. Especially problematic is the fact that NDT inspection often involves a number of parties, like the inspection system manufacturer, the inspection company, and the owner of the inspected installation, not all of them sharing the same interests. Successful introduction of the automated NDT data interpretation would have to involve cooperation of all the parties, which is not easy to realise. Therefore, introduction of automatic data interpretation is easier when fewer parties are involved, for example, when the developer of the inspection system also does the inspections. Still, whenever two of more interested parties are involved the economic analysis of the possible benefits resulting from automation becomes more difficult.

The problem is made even greater by the presence of many rules regulating the NDT inspection (prescribing the inspection intervals and the inspection technique). They make analysing possible advantages more difficult. Moreover, keeping with the regulations gives the companies a feeling of security and discourages from making any changes. Very common is the attitude: "everybody does it like this, why should we make any changes."

As could already have been noticed, the problems in automating NDT inspection are more organisational and economical than related to unavailability of suitable techniques for data interpretation. Certainly, AI provides a large number of suitable techniques, of which CBR is possibly the one most generally applicable to NDT problems. The increasing power of computer hardware should make CBR even more attractive, and increasing computerisation of NDT inspection should make the automation of the NDT data interpretation easier.

8.6 Conclusions

This thesis has presented the issue of automating NDT data interpretation. It has been shown that because of a number of reasons, like, for example, the character of data to be interpreted, high requirements on the reliability, and economic viability of the interpretation systems, several well-known data-interpretation techniques, like for instance statistical classifiers, ANNs, and expert systems, may be not suitable for practical NDT applications. On the other hand case-based reasoning has a number of advantages which make it suitable for NDT data interpretation in practice. Some of the advantages, like ability to learn, high reliability, lower development and maintenance costs are especially important in this context.

Within the ANDES project two CBR systems for NDT have been developed: one for the interpretation of eddy-current data and another for the interpretation of ultrasonic B-scans. The experiences with the whole development process, as well as the results of the tests conducted on real field data, confirm the theoretical considerations about the suitability of CBR in practice. The prototype systems were successful in handling complex data, there was no need to gather extensive data sets for training, and only general domain knowledge was required to develop the systems. Nevertheless, the recognition percentages ranged from satisfactory to high, and the reliability was high. A major disadvantage of CBR, the slow retrieval speed, should loose on significance because of the advances in computer hardware.

Altogether, it can be concluded that CBR is certainly a suitable technique for the interpretation of NDT data, a technique which for many practical problems may be the technique of choice, either used on its own or in combination with other AI techniques. Building on the results of the ANDES project, at TNO-TPD work is being done on an implementation of the CBR system for the URS inspection train. This should provide us with the insights how CBR fares in everyday NDT practice, which will be valuable for the development of any future CBR systems for NDT data interpretation.

Appendix A: Computer hardware prices

Two of the disadvantages of CBR are: the size of the case-base that has to be maintained and, a related problem, the long retrieval times. Within the CBR field, much research has been devoted to the problem of speeding up the retrieval times. However, it seems that within the next few years this problem will "solve itself" due to the increasing power and decreasing prices of the computer hardware. The development of hardware prices over a period of the last four years (beginning from January 1994) is illustrated in this appendix. The prices of the hardware are in DM and have been taken from the advertisements placed in *c't - computer magazine*, thus, these are real market prices (there are some fluctuations caused by the changes in DM/$ exchange rate).

Extrapolating the CPU power (in MHz) from the graphs shown below is difficult because of the different processor generations. In August 1998 the Pentium processor was discontinued, but it can be extrapolated that in August 1999 an affordable computer will have a CPU comparable to a 500 MHz Pentium. As far as Pentium II is concerned, a prediction can be made that at the same time the price of PII 450 Mhz CPU will fall down to about DM 300,-.

Predicting memory prices is very difficult, but if the recent stagnation does not continue then at the end of 1999 128MB RAM should cost about DM 100,-. Hard disk prices are much easier to predict and for the typical price of DM 350,- it should be possible to buy 16GB hard disk at the end of 1999.

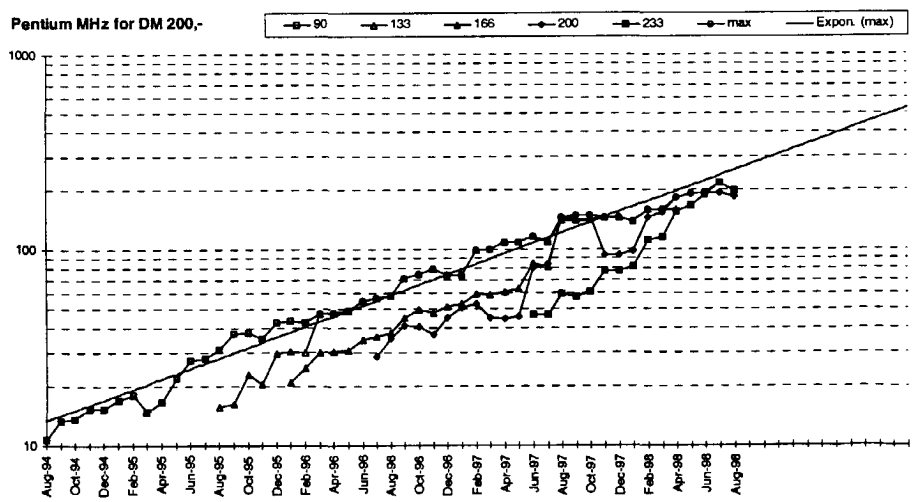


Figure A.1: Speed of an imaginary Pentium processor available for DM 200,-. The amount of MHz power (in the low-end range) one can afford for the same price increases per year with almost exactly a factor of 2.

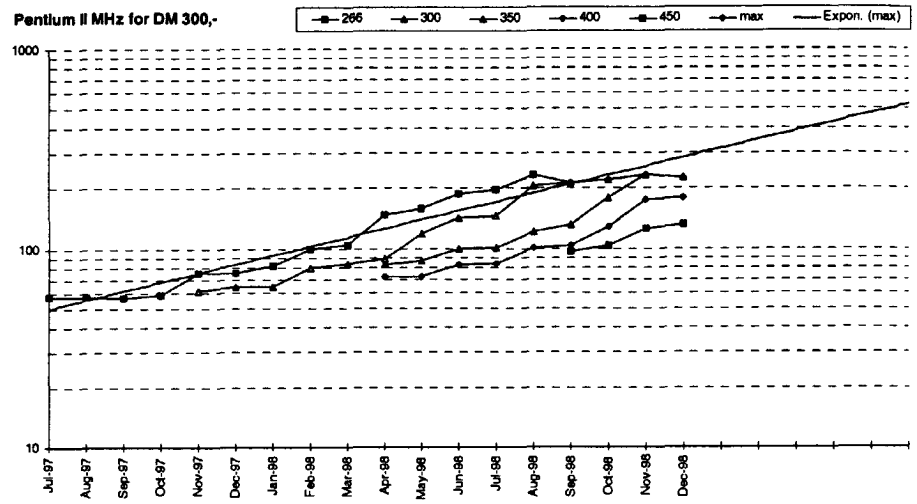


Figure A.2: Speed of an imaginary Pentium II processor available for DM 300,-. The amount of MHz power (in the high-end range) one can afford for the same price increases per year with about a factor of 2.

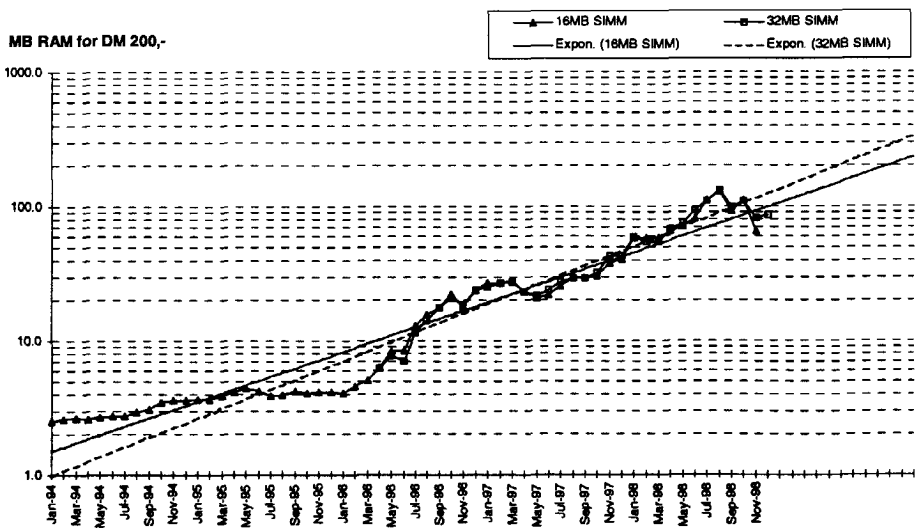


Figure A.3: Amount of RAM one can afford for DM 200,-. Until the beginning of 1996 the prices of RAM memory stagnated. Since then prices have kept falling with a factor of about $^1/_3$ each year, until the 'Asian crisis' in Summer 1998.

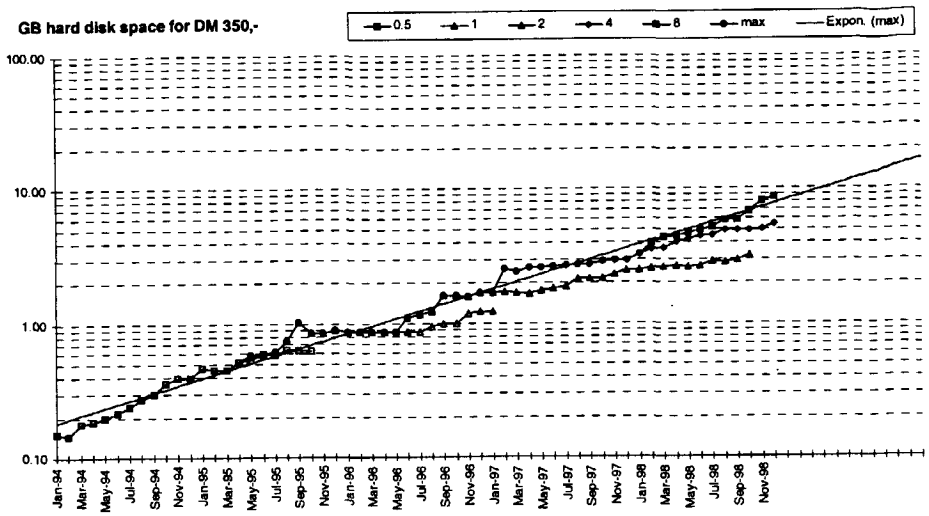


Figure A.4: Amount of hard disk space available for DM 350,- (the data is for hard disks with capacity from 0.5GB to 8GB). On average the amount of hard-disk space one can afford for the same price rises with a factor of 2 each year.

Appendix B: Publications

Based on the research presented in this thesis the following reports and papers have been published. They are listed in the order they were written (not published) to reflect the progress of the project.

Jarmulak, J., *First semi-annual report Andes project: Literature survey*, TNO-report, FSP-RPT-950033, 16 May 1995.

Jarmulak, J., *Second semi-annual report Andes project: Rule-base (URS) an CBR (EC) prototypes*, TNO-report, FSP-RPT-950090, 29 November 1995.

Jarmulak, J., "A Method of Representing and Comparing Eddy Current Lissajous Patterns", *Review of Progress in Quantitative Nondestructive Evaluation*, Vol. 16, D.O. Thompson and D.E. Chimenti (eds.), Plenum Publishing, New York, 1997, pp. 303–308.

Jarmulak, J., "B-scan Image Clustering and Interpretation in Ultrasonic Rail-Inspection System", *Proceedings of the second annual conference of the Advanced School for Computing and Imaging, Lommel, Belgium, June 5-7*, 1996, pp. 190–195.

van't Hoff, P., *Design of a case-base for Eddy Current CBR system: optimal case representation and retrieval*, TNO-report, FSP-RPT-960101, 1996.

van't Hoff, P., Jarmulak, J., Kerckhoffs, E.J.H., and van't Veen, P.P., "Methods for Transformation Invariant Representation and Comparison of Closed Curves", *ASCI'97 Proceedings of the third annual conference of the Advanced School for Computing and Imaging, Heijen, the Netherlands, June 2-4*, 1997, pp. 167–173.

Jarmulak, J., van't Veen, P.P., and Kerckhoffs, E.J.H., "Case-Based Reasoning in an Ultrasonic Rail-Inspection System", *Case-Based Reasoning Research and Development: Second International Conference on Case-Based Reasoning, ICCBR-97, Providence, RI, USA, July 1997, Proceedings*, D.B. Leake and E. Plaza (eds.), Springer-Verlag, Berlin, 1997, pp. 43–52.

Jarmulak, J., "Case-Based Reasoning for Automatic Interpretation of Data from Eddy-Current Inspection", *Review of Progress in Quantitative Nondestructive Evaluation*, Vol. 17A, D.O. Thompson and D.E. Chimenti (eds.), Plenum Publishing, New York, 1998, pp. 767–774.

Jarmulak, J., *LISSA - a system for eddy-current data interpretation using case-based reasoning*, TNO report, HAI-RPT-970055, 2 September 1997.

Jarmulak, J., *Case-based reasoning for interpretation of ultrasonic rail-inspection data*, TNO report, HAI-RPT-970070, 27 October 1997.

Jarmulak, J., *Final report ANDES project: Case-based reasoning for NDT data interpretation*, TNO report, HAI-RPT-970078, 26 November 1997.

van't Veen, P.P. and Jarmulak, J., "Automated Interpretation of Heterogeneous NDT Measurements", *7th ECNDT Proceedings*, Vol. 3, pp. 2355–2362.

Jarmulak, J., Kerckhoffs, E.J.H., and van't Veen, P.P., "Hybrid knowledge based system for automatic classification of B-scan images from ultrasonic rail inspection", *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98) - Tenth Conference on Innovative Applications of Artificial Intelligence (IAAI-98)*, 1998, pp. 1121–1127.

Jarmulak, J., "Case-based classification of ultrasonic B-scans: case-base organisation and case retrieval", *Advances in Case-Based Reasoning: EWCBR-98 Proceedings*, B. Smyth and P. Cunningham (eds.), Springer-Verlag, 1998, pp. 251–261.

Subject index

Numerics

1-D scan 26
2-D scan 26
3-D scan 26
4 REs (CBR cycle) 58

A

absolute probe 120
accessibility problem 68
acoustic
  (vibration) monitoring 17
  emission 1, 7, 16
  impact 7, 17
  monitoring 7
  NDT methods 16
acquisition system 5
adaptation 73
  by substitution 74
  by the system 74
  by the user 74
  case-based 74
  generative 74
  knowledge 61
    learning 62
  transformational 74
AI 2, 33, 34, 35, 91
  applications 51
  connectionist 36
  knowledge reuse 52
  strong 35
  symbolic 36
  two paradigms 35
  weak 35
alumino-thermic weld 160
ANDES 2, 85, 119, 159
ANN 29, 36, 39, 47, 49, 50, 51
  advantages 39
  classifier 98, 108
  for modelling 91
antecedent (of a rule) 38
anti-vibration bar 119, 127
ARCHIE 83
array probe 121
ART*Enterprise 85
ART1 44
Artificial Intelligence - see AI
artificial neural networks - see ANN
A-scan 13, 95
  interpretation 15
ASME
  calibration tube 122, 143
assumption
  closed-world 38
  regularity of the world 56, 182
autoassociative network 42
automated
  characterisation 28
  classification 28
  detection 27
  interpretation 27
  scanning 26
    benefits 27
averaging 14

B

backpropagation (of error) 40
backward chaining 38
baffle 119
  defects under 130
  plate 127
  signal from 127
BAM 42, 43
  capacity 43
Bayes theorem 46
Bayesian
  belief networks 46
  classifier 98
belief networks - see Bayesian b. n. 46
Bidirectional Associative Memory - see BAM
blackboard system 93
blind tests 21
bobbin coil probe 11, 120
BOLERO 83
Boltzman machine 42
brain 36
B-scan 13, 162
  image 28
  interpretation 15
B-spline 97

C

C4.5 47, 180
calibration 6, 28, 121, 138
  curve 28, 29, 122, 145
  tube 122
    ASME 122
case 66
  adaptation 57, 73
  competence 81
  engineering 79
  features 60
  knowledge contained in 61
  matching 69
    in LISSA 139
    in URS 177
  problem coverage 66
  representation 66
    formalisms 68
  retrieval 57, 70, 72
    in URS 182
    parallel 73
  storage 76
  visualisation 68
  vocabulary 60
case-base 45
  large 64
  maintenance 80, 186
  management 81
  organisation 61
    clustered 70
      in URS 182
    flat 70
    hierarchical 70
    in LISSA 138
    in URS 179
    networked 71
  updating 57
case-based reasoning - see CBR
CaseLine 84
CASEY 83
CASUEL 68, 86
cavitation 119
  damage 125
CBR 45, 48, 49, 51, 55, 159
  advantages of 62
  basic assumptions 56
  disadvantages of 64
  for design 83
  for diagnosis 83
  for help desks 83
  for interpretation 58, 83
    of data 105
  for planning 83
  for problem solving 57

in LISSA 138
in URS 172
instance-based 63
learning 62
paradigm 84
types of knowledge used 60
working cycle 57
4 REs 58
CBR-Express 69, 85
CBR-TFS 83
CBR-Works 86
certainty factors 45, 47
characterisation 6
automated 28
CHEF 83
chemotaxis 41
Chinese Room 35
robot argument 35
system argument 35
classification 6, 93, 97
automated 28
rule based 168
classifier
(k-)nearest neighbour 98
ANN 98, 108
Bayesian 98
comparison 98
minimum-mean-distance 98
MLP 41
Parzen-window 98
POLYCLASS 99
problems with 100
CLAVIER 63, 84
CLIPS 163, 168, 169, 175, 176, 184
closed-world assumption 38
Cluster Discovery Network - see ART1
clustering 95
fuzzy 96
hard 96
K-means 95
leader algorithm 95
minimum-spanning-tree 95
of URS images 170
possibilistic 96
cognitive science 34
computation 35
condenser 119
conjugate-gradient minimisation 41
connectionist
AI 36
paradigm 36, 39
consciousness 34, 35
consequent (of a rule) 38
convolution 14
corrosion 128
couplant 13
coverage (of NDT inspection) 21
crack 119, 124
kidney-shaped fatigue 159
C-scan 13
image 28
interpretation 15
CuNi 143
curvature 97
Cybenko 40
CYC 39
Cycorp 39
CYRUS 56

D

data
acquisition
eddy current 9
conditioning 91
from ultrasonic testing 13
inhomogeneous 2
interpretation 91, 92
eddy current 10
methods 93
schemes 93
preprocessing 28, 94
presentation
eddy current 10
decision tree 47, 98, 180
declarative program 35
deconvolution 14, 18, 28
deduction 37
defect
in heat-exchanger tube 119
in rails 159
sizing 6
Déjà Vu 83
delta rule 40
generalised 40
Dempster-Shafer theory 45, 46, 47, 104
DENDRAL 37
dent 119, 126, 130
design (CBR for) 83
destructive testing 1
detection 5, 135
automated 27
diagnosis (CBR for) 83
differential probe 120
diffraction (wave) 12
digital equipment 6
distance
Euclidean 69
Hamming 42
drift (in signal) 27, 128
removal 135

E

EADOCS 83, 86
Easy Reasoner 85
eddy current 1, 7, 8, 120
applications 8
data
acquisition 9
interpretation 10
presentation 10
typical responses 123
frequency 121
indications
combinations of 129
from artefacts 126
from construction elements 127
from defects 124
from noise 127
from non-defects 126
mixing 10, 28
penetration depth 8
probe 120
absolute 120
array 121
bobbin coil 11, 120
differential 120
rotating 121
signal processing 9
electric current (NDT techniques) 7
electromagnetic-electronic (NDT techniques) 7
ELIZA 34
emission counts 17
emotions (human) 33
endoscope 120
error
backpropagation 40
Euclidean distance 69
evaluation (in CBR) 75
expert system 37, 39, 49, 63, 108
advantages 38
disadvantages 38
for data interpretation 103
fuzzy 46
shell 39
working of 37
expertise (of NDT inspector) 28

F

fact (in expert system) 37
failure
mechanics 26
probability 23
feature extraction 93, 94
for shapes and curves 96
FFP 25, 27
filter
adaptive 94
in frequency domain 94

lowpass and bandstop 134
median 134
first-order
logic 37, 46
minimisation 41
fish-plated joint 159
fitness for purpose - see FFP
forward-chaining 38
Fourier
coefficients 97, 140
descriptors 97
transform
short-time (STFT) 18
frames 38
Freeman chain code 97
frequency spectrum
analysis 17
fuzzy
expert system 46
logic 46
matching 141
reasoning 47
set 46

G

gamma
radiation 18
radiography 7
generalisation 63
genetic algorithms (GA) 48
Giarrtano 37
Gödel's theorem 36
gradient descent 40, 41
GREBE 83

H

Hamming
distance 42
classifier 43
net 43
heat exchanger 119
inspection 119
methods 120
using EC 2, 119
tube 119
defects in 119
help desks (CBR for) 83
heuristics 94
hidden layer 40
histogram of matches 142
HOMER 84, 86
Hopfield
net 42
capacity 43
Hough transform 96, 171
hybrid systems 106
HYPO 83
hypothetical reasoning 38

I

ICEPAK 95
ID3 47
if-then rules learning 47
ILP 47
imitation game (Turing test) 34
impedance 9
Inconel 119
incremental learning 65, 77
index choice (automatic) 69
indexing 68
inductance 8
inductive logic programming - see ILP
inference
cycle 37
engine 37
inference engine 37, 39
INRECA 66
inspection
added value 24
benefit 23
cost 24
coverage 21
instance-based learning 48
intelligence 33
artificial 33
behaviouristic definition 33
human 34
test 33
interpretation
automated 27
CBR for 83
inversion 91
of NDT data 6
IRIS 124

J

JUDGE 83
JULIA 58, 83

K

Keiser effect 16
KICS 83
k-nearest neighbour classifier 98
learning 48
knowledge
accumulation in CBR systems 64
acquisition 51, 63
base 37
-based system 34
containers in CBR 60
contextualised in CBR 63
engineering 34
in CBR 79
representation formalisms 38
Kohonen
network (vector quantisation) 44
self-organizing feature map 44

L

lazy
learning 48
problem-solving 65
benefits 65
leader algorithm (for clustering) 95
learning
CBR 62
incremental 62, 65, 77
instance-based 48
on-line 62
reinforcement 48
Levenberg-Marquardt minimisation 41
linear discriminant function 98
liquid penetrant 7
LISSA 67, 69, 83, 119
calibration 138
case description 141
CBR 138
overview 132
preprocessing 133
reliability 154
speed of operation 153
test results 142
user interface 142
Lissajous
curve 10, 140
comparison 140
Loebner Prize 35
logic
first-order 37, 46
fuzzy 46
Logic Theorist 37
logical reasoning 37

M

Machine Learning - see ML
magnetic
flux leakage 7, 12, 120
induction (principle) 8
particle 7
saturation 11
magnetic inclusions 126
magnetite deposits 126
maintenance 50, 52
of CBR systems
ease of 63
MAXNET 43

McCulloch, Warren 34, 36
measurements in images 28
median filter 134
memory organisation packet - see MOP
MFL - see magnetic flux leakage
microwave radiation 7
minimisation
  first-order 41
  second-order 41
minimum spanning tree - see MST
mixing 10, 94, 137
ML 47, 62
MLP 40
  classifier 41
  training 40
mode conversion (wave) 12
moment invariants 97
momentum term (in MLP training) 40
MOP 55
  example 55
MST 95, 172
multifrequency (eddy current) 9
multilayer perceptron - see MLP
MYCIN 37, 45, 83

N

NDT 1, 5
  certification 113
  data
    characteristics 107
    interpretation 91
  inspection
    chain 5
    performance 20
  interpretation of data 5
  standards 113
  technique 5, 6
nearest-neighbour retrieval 72
network
  autoassociative 42
neural networks (artificial) 36
neuron 36
  simulator 34, 36
noise 67
  in CBR 64
  in EC inspection 127
  model 94
  modelling 28
  removal 28
non-computable system 36
nondestructive testing - see NDT

O

offset (in signal) 27, 128
  removal 134
optical NDT techniques 7

P

pattern recognition 28, 93, 94
  pipeline 93
penetration depth (of eddy currents) 8
perceptron 40
PFI 22, 26, 197
pilgrim noise 94, 128
PISC 21
pitting 119, 124
Pitts, Walter 34, 36
planning (CBR for) 83
POD 21, 26, 197
POFA 22
probability
  of detection - see POD
  of false indications - see PFI
probe
  absolute 120
  bobbin coil 120
  differential 120
  wobble 128
problem
  description 66
    parametric 67
    raw 66
  non-linearly separable 40
  solution 67
PROTOS 59, 62, 69, 71, 83
pulse-echo ultrasonics 12
push-puller 9, 26

Q

quality of NDT 20
quantum effect 36
quickprop 41
Quinlan 47

R

radial basis function network - see RBFN
radiation NDT technique 7, 18
radioactive isotopes 18
rail inspection 159
  using ultrasound 2
rails
  defects in 159
  manufacturing faults 159
RBFN 41
  training 42
reasoning
  cycle 38
  fuzzy 47
  hypothetical 38
  logical 37
  symbolic 35
  with uncertainty 45
ReCall 86
reflection (wave) 12
regularity of the world assumption 56, 182
reinforcement learning 48
reject threshold 100
reliability 49, 63, 92, 111
  of detection 22
  of LISSA 154
  of NDT inspection 21
  of URS-CBR 192
ReMind 86
remote field eddy current 11, 120
reporting 6
resistance 8
retrieval 72
  in CBR 61
  incremental 73
  nearest-neighbour 72
  parallel 73
RFEC 11
Riley 37
ROC 22
Rosenblat, Frank 40
rotating probe 121
rule 37
  antecedent 38
  -based classification 168
  consequent 38

S

S3-CASE 85
saturated eddy current 120
scanning
  automated 26
    benefits 27
Searle (Chinese Room) 35
second-order minimisation 41
self-organizing feature map - see SOM
semantic nets 38
sensitivity (of NDT inspection) 20
sensor 5
short-time Fourier transform (STFT) 18
SHRDLU 37
signal
  filtering 14
  processing
    in ultrasonics 14
  thresholding 91
silver bullet 87
similarity
  "useful" 69

measure 61, 69
situation assessment (in CBR) 61, 73
skin effect 8
solution
adaptation (in CBR) 73
evaluation (in CBR) 57
SOM 44
sonic-ultrasonic NDT techniques 7
soul 36
speed
of AI system 50
of CBR system 80
of NDT inspection 20
Spock (Mr.) 33
statistical techniques 50
steam generator 119
stereography 18
strong AI 35
claims 35
false 35
subsymbolic representation 39
supervised training 40
SWIFT 85
symbol 35
manipulation 35
symbolic
AI 36
paradigm 35, 37
reasoning 35

T

Tchebycheff coefficients 97
thermal
imaging 7
NDT technique 7, 19
static 19
transient 19
thermit weld 160
threshold estimation 142
thresholding 91
training
MLP 40
RBFN 42
supervised 40
unsupervised 40
transformation invariance 69
transmission ultrasonics 13
triangulation (of acoustic emissions) 17
trigram 69
tube sheet 119, 127
Turing 36
Alan 34
Machine 35
test 33, 34

U

ultrasonic
rail inspection 2
wave 12
ultrasonics 7
ultrasound 1, 12
uncertainty 50
in CBR 64
reasoning with 45, 104
unsupervised training 40
URS 159, 160
CBR 172
second generation 160
transducer assembly 161
user interface
of AI systems 52
of CBR systems 78

V

vector quantisation 44
virtual reality (VR) 68
visual inspection 120

W

wavelets 97
weak AI 35
claims 35
weak-theory domains 63
Weizenbaum, Joseph 34
Widrow–Hoff rule 40
Wigner-Ville distribution 18
Winograd 37

X

X-radiography 7
X-ray 1
image 28
microfocus tubes 18
NDT technique 18
stereography 18
tomography 19

Z

Zernike moments 97

Author index

A

Aamodt 48, 53, 58, 59, 62, 63, 87
Abelson 55, 89
Adler 15, 29
Aha 58, 65, 66, 77, 79, 87, 105, 114
Albert 87
Aleksander 40, 53
Allen 115
Althoff 48, 53, 66, 82, 85, 86, 87
Asada 97, 114
Ashworth 157
Auriol 87

B

Baba 41, 53
Ballard 96, 114
Bareiss 89, 116
Barletta 87
Barrodale 31
Barsky 115
Basart 114, 117
Baudrillard 97, 102, 103, 114
Bauman 41
Baur 53
Bengio 42, 53
Benoist 28, 29, 30, 94, 103, 114, 128, 134, 157
Bergmann 68, 87, 88, 89
Berry 102, 114
Bieth 21, 29
Birac 29
Birks 12, 29
Bonzano 58, 88
Bose 115
Bouchon-Meunier 46, 53
Bouman 53
Bowker 119, 123, 157
Brady 97, 114
Bräuer 87
Bray 160, 195
Brooks 87
Brousset 97, 102, 103, 114
Brown 30
Brudnoy 115

C

Carkhuff 96, 98, 99, 103, 114
Cecco 8, 29, 121, 157
Chan 108, 114
Chapman 99, 102, 103, 114
Chen 42, 53, 100, 114, 167, 195
Cherpentier 114
Chiou 22, 29
Chuang 92, 114
Cohen 38, 53
Comby 29
Conruyt 87
Cornwell 116
Coulon 67
Cowan 53, 114
Craw 62, 74, 83, 88
Cunningham 74, 89
Cybenko 40, 53

D

d'Annucci 158
Damgaard Kristensen 30
Dau 11, 30
David 114
Davis 33, 36, 53
De Mori 42, 53
Dearden 78, 88
Dekking 114, 140, 158
Dittrich 87
Doctor 99, 114
Downs 28, 30
Duchene 30
Dumer 53
Dunlop 102, 115

E

Engelmore 93, 114
Englund 115
Eser 158
Esveld 160, 195

F

Fahr 114
Faur 28, 30
Feigenbaum 38, 53
Fisher 65, 88
Flandrin 18, 30
Flannery 158
Freg 96, 115, 171, 195
Frigui 115
Fu 98, 114, 137, 140, 158

G

Gaillard 29, 114, 157
Georgel 103, 114
Giarratano 45, 53
Gierl 70, 89
Giling 162, 171, 195
Göker 78, 79, 81, 84, 86, 88
Gomonov 102, 103, 115
Gonzalez 96, 114, 171, 195
Grant 53, 114
Green 29
Grimaldi 44, 53

H

Hagemaier 9, 30
Hallam 115, 195
Hameroff 36, 53
Hammar 19, 30
Hanney 88
Hanratty 53
Hansen 30
Harrington 99, 114
Harrison 105, 114
Hartigan 95, 115
Hartog, den 95, 115
Havira 167, 195
Hay 30, 114, 115
Helfman 51, 53
Hellyar 94, 115
Hendler 73, 88
Hinkle 88
Hinton 40, 54
Hoff, van't 97, 115, 139, 158, 207
Holte 89, 116
Hopgood 96, 105, 106, 107, 115, 171, 195
Hu 97, 115
Hudson 115

I

Ifeachor 115
Ingham 53

J

Jacobs 41, 53
Jansen 168, 195
Jarmulak 97, 115, 152, 153, 154, 158,

169, 172, 179, 195, 207
Jungman 29

K

Kalyanasundaram 116
Kamgar-Parsi 43, 53, 96, 115
Kang 97, 103, 106, 115
Kashyap 83, 89
Katwijk, van 159, 195
Keane 88
Keller 96, 115
Kerckhoffs 3, 31, 117, 158, 195, 207
Kettler 73, 88
Khotanzad 97, 98, 100, 115
Kibler 87
Kinley 88
Kohonen 44, 53
Koller 46, 53
Kolodner 56, 57, 58, 62, 66, 68, 69, 80, 83, 84, 88, 182, 195
Kooperberg 99, 115
Kraats, van de 52, 53
Krishnapuram 96, 115, 171, 195
Krzywosz 11, 30
Kwon 97, 103, 106, 115

L

Lacasse 95, 115
Laine 97, 117
Leake 57, 60, 74, 80, 88
Lee 90
Lesiak 167, 195
Lesselier 30
Levy 103, 104, 115
Lim 99, 115
Loh 115
Loomis 115
Lopez 83, 88
Lord 28, 30, 94, 95, 97, 98, 99, 102, 103, 115, 116, 140, 158
Lorenz 2, 3, 31, 117
Lu 97, 98, 100, 115
Lunin 102, 103, 115

M

MacDonald 114
Maciga 29
MacIntyre 105, 116
Mack 113, 116
Maes 52, 53, 74, 88
Malek 70, 88
Manago 87
Mapps 115
Marir 45, 54, 83, 84, 85, 89
Mark 62, 74, 79, 84, 88
Martinez 69, 89
Maszi 91, 94, 116
Matthews 114
Maurer 87
McIntire 29, 31
McKenna 81, 89
McMaster 31
McNab 102, 106, 115
McRae 31
Mitchell 47, 54
Monebhurrun 11, 30
Montague 41, 54
Morgan 93, 114
Morisseau 30
Morizetmahoudeaux 29, 114, 157
Morris 54
Morton 40, 53
Moulder 114, 117
Moulton 51, 54
Murtagh 96, 116
Mustafa 30

N

Nagy 29
Nair 42, 54, 99, 103, 116
Nasraoui 115
Netten 83, 86, 89
Notvest 30

O

Oatley 105, 116
Oksman 30
Okure 22, 30
Oliveira 68, 89
Oppenheim 95, 116
Oppenlander 115
Otterloo, van 97, 114, 140, 158
Ousterhout 86, 89

P

Pantleon 88
Papadakis 113, 116
Pelletier 114
Penrose 36, 53, 54
Perner 67, 89, 105, 116
Persoon 137, 140, 158
Peshkin 22, 30
Peterson 28, 30
Pfisterer 121, 158
Picton 115, 195
Pigeon 29, 114, 157
Plaza 58, 59, 83, 87, 88
Poetz 120, 158
Pollock 30
Porter 59, 62, 69, 75, 83, 89, 105, 116
Post 183, 194, 196
Pratt 17, 30, 100, 103, 116
Press 138, 158

Q

Quinlan 47, 54, 71, 89, 98, 116, 180, 196

R

Raj 116
Rajagopalan 104, 106, 116
Richter 60, 89
Riesbeck 45, 54, 56, 83, 89
Riley 45, 53
Robertson 90
Roos 160, 166, 168, 169, 196
Rose 29
Roskies 97, 117
Roth-Berghofer 88
Rowe 88
Roy 30
Rumelhart 40, 54

S

Sansalone 17, 30, 100, 103, 116
Sarle 44, 50, 54
Satish 28, 30, 94, 102, 103, 115
Schafer 95, 116
Schaffer 98, 100, 116
Schalkoff 98, 116
Schank 45, 54, 55, 56, 83, 89
Schlimmer 65, 88
Schmidt 70, 89
Searle 35, 54
Sharp 29, 157
Shih 115
Shmerr 29
Siering 183, 194, 196
Simoudis 88
Sladojevic 159, 196
Sloman 36, 54
Smith 10, 30
Smyth 74, 81, 83, 89
Spanner 16, 30
Stahl 68, 87
Stepinski 91, 94, 97, 116, 140, 158
Stiller 194, 196
Stolte 10, 30
Stone 115
Surma 76, 81, 170, 193, 196

T

Tait 105, 116
Tange 168, 171, 196
Teukolsky 158
Thompson 29, 58, 89
Toitgans 88

Traphöner 87, 88
Tsatsoulis 83, 89
Turing 34, 48, 54, 64, 89
Tyburcy 76, 81, 193, 196

U

Udpa 30, 54, 95, 96, 97, 98, 99, 102, 103, 114, 116, 140, 158
Upadhyaya 94, 117

V

Van Drunen 29, 157
Van Trees 22, 30
Vanhoof 170, 196
Veen, van't 158, 195, 207
Vesth 26, 30
Vetterling 158
Vincent 88
Vollrath 89

W

Waas 158
Wall 20, 21, 30
Wang 42, 44, 54, 91, 117
Warnes 157
Watrous 41, 54
Watson 45, 54, 68, 83, 84, 85, 89
Wedgwood 20, 21, 30
Wenk 7, 31
Werf, van der 159, 196
Wess 87, 88
Wielinga 2, 3, 28, 31, 100, 117
Wilke 60, 62, 88, 89
Williams 40, 54
Willis 54
Wilson 69, 80, 88, 89
Wiratunga 88
Woodcock 115, 195
Woods 96, 114, 171, 195
Wuch 97
Wunsch 117
Wybo 88

Y

Yan 94, 117
Yang 83, 90

Z

Zadeh 46, 54
Zahn 97, 117
Zala 14, 31
Zorgati 30, 103, 114
Zurada 40, 54

Curriculum Vitae

PERSONAL

name:	Jacek Jarmulak
born:	2 June 1967, Zgorzelec, Poland

EDUCATION

9/81 – 5/85	Matura, Liceum Ogólnoksztalcace im. M. Kopernika in Lubin, Poland.
9/85 – 6/87	Two years of five-year studies in Mechanical Engineering completed at Wroclaw University of Technology, Poland.
9/90 – 9/94	M.Sc. in Technical Informatics from Delft University of Technology, The Netherlands. Thesis title "NeuroControl Workbench."

WORK EXPERIENCE

11/87 – 9/89	Work in a kibbutz (Ashdot Ya'akov Meuhad) and moshav (Neve Yamin) in Israel.
2/95 – 10/98	Research Assistant at Delft University of Technology, Faculty of Information Technology and Systems, Department of Technical Mathematics and Informatics, Delft, The Netherlands.
11/98 – present	Research Assistant at The Robert Gordon University, School of Computer & Mathematical Sciences, Aberdeen, Scotland.