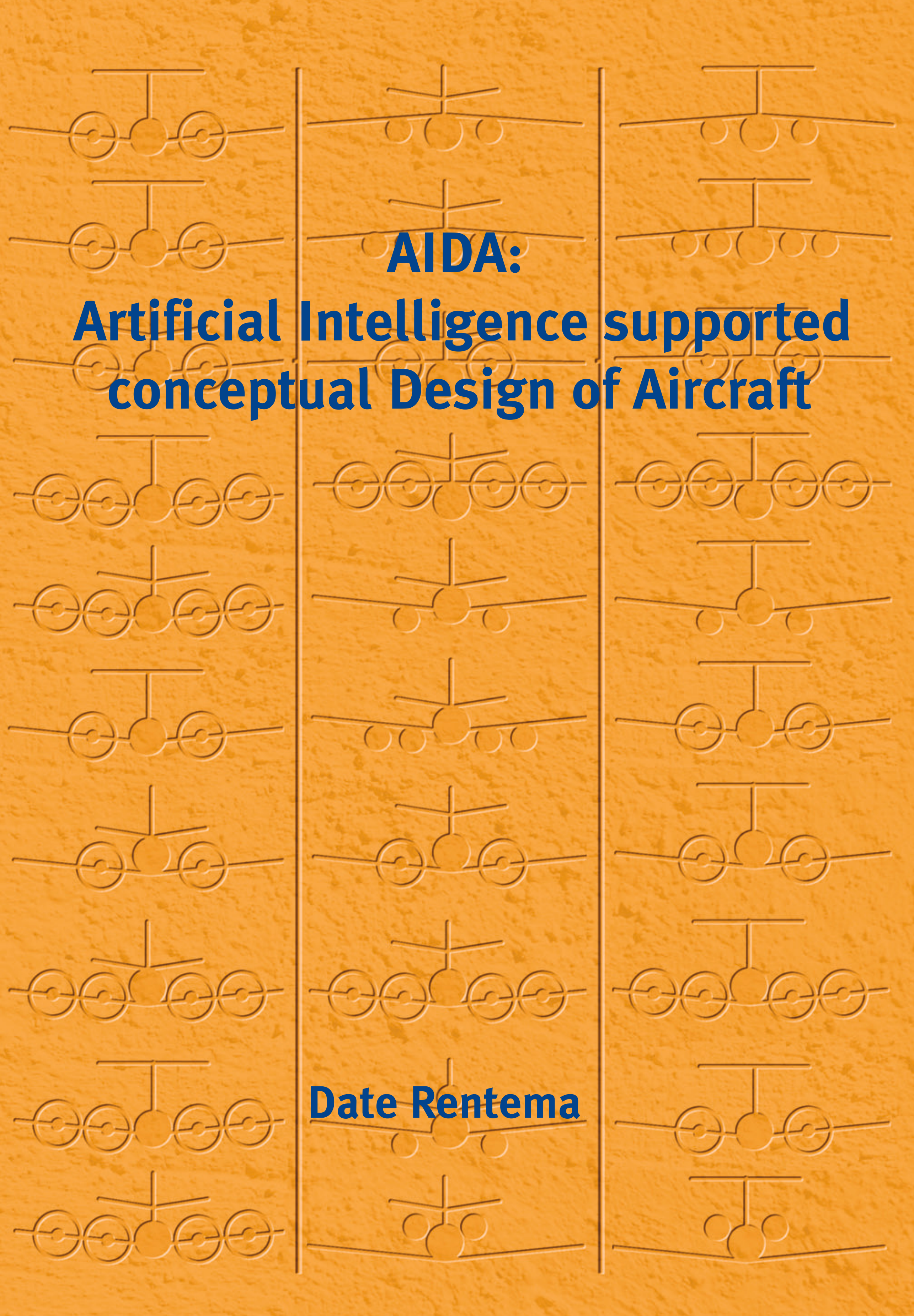


AIDA: Artificial Intelligence supported conceptual Design of Aircraft

Date Rentema

AIDA:
Artificial Intelligence supported conceptual Design of Aircraft

Proefschrift

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus prof. dr. ir. J.T. Fokkema,
voorzitter van het College van Promoties,

in het openbaar te verdedigen op maandag 15 november 2004 om 13:00 uur

door Date Willem Egbert RENTEMA
ingenieur Luchtvaart en Ruimtevaart
geboren te Soest

Dit proefschrift is goedgekeurd door de promotoren:
Prof. dr. ir. F.W. Jansen
Prof. dr. ir. E. Torenbeek

Samenstelling promotiecommissie:

Rector Magnificus,	voorzitter
Prof. dr. ir. F.W. Jansen,	Technische Universiteit Delft, promotor
Prof. dr. ir. E. Torenbeek,	Technische Universiteit Delft, promotor
Prof. dr. H. Koppelaar,	Technische Universiteit Delft
Prof. dr. ir. M.J.L. van Tooren,	Technische Universiteit Delft
Prof. dr. F.M.T. Brazier,	Vrije Universiteit Amsterdam
Prof. dr. J. Fielding,	Cranfield University (UK)

to my dear mother
and the memory of my father

Summary

This thesis describes the development of a computer support tool that supports the initial, conceptual design process. In this first design phase one or more concepts are defined which are assumed to be able to comply with the design specifications. These concepts can be elaborated in more detail with the aid of the conventional CAD tools that have been developed in the last decades and are currently widely available. Due to the non-deterministic character of conceptual design, however, such conventional computer techniques are inadequate to support this phase. Therefore the use of Artificial Intelligence techniques has been investigated in this project.

The initial application was chosen in the area of aircraft design, because of its complexity. The tool and methodology are, however, also applicable to support the design of ships, buildings and industrial appliances.

The developed design tool and methodology is based on the well-known "design cycle", which in our version consists of four steps:

1. *Suggesting* a concept: this task is supported with Case-Based Reasoning (CBR) techniques. From a case-base that is filled with data of existing artefacts, one artefact is selected whose performances best match the design specifications. This case is used as a starting point and delivers the configuration or topology of the design, and the initial parameter settings. Possibly two or more cases can be combined, resulting in an *adapted* case.
2. *Simulating* the artefact's performances: this task is supported with Rule-Based Reasoning (RBR) and Geometric Modelling (GM) techniques. With the rules representing the domain knowledge in an explicit, algebraic format, a network of rules and relations is built that relates the sizing parameters with performance parameters. GM is used to build a geometric model and to handle the geometric constraints.
3. *Evaluating* these performances: using the RBR network of relations and the GM constraints, the related solvers allow quickly and easy changing of parameter values to modify the design.
4. *Proposing modifications*: the design tool does not support this task explicitly. However, as is mentioned in step 1, a first modification of the case's topology (*adaptation*) is supported with CBR. Then the cycle is repeated until the design concept is satisfactory and can be used as input for further elaboration and evaluation with existing CAD/CAE tools.

Each of these three techniques, CBR, RBR and GM, has been implemented in a separate module. Existing programs have been used and adapted for this study.

To test the set-up and viability of the design tool, a case-study has been performed, concerning the conceptual design of an 80 passenger commercial

aircraft. The study has proven that the design approach is useful for "configuration design" type of design tasks, but is less suitable for innovative and creative design.

Although each technique has shown to be suitable for its task, there are also some difficulties. For example, each technique requires substantial effort in the pre-processing phase, when filling the data or knowledge bases. And experience is needed to operate the CBR and RBR modules properly. Many apparently minor practical issues can have a serious effect on the efficiency of the techniques.

Issues which need much attention in following development studies are case adaptation and integration of the three applied techniques.

Samenvatting

Dit proefschrift beschrijft de ontwikkeling van een computer gereedschap dat het initiële, conceptuele ontwerp proces ondersteunt. Het doel van de eerste ontwerpfase is om één of meerdere concepten te definiëren waarvan verondersteld wordt dat ze voldoen aan de ontwerp specificaties. Deze concepten kunnen uitgewerkt worden m.b.v. conventionele CAD ("Computer Aided Design") gereedschappen die in de laatste decenia zijn ontwikkeld en algemeen beschikbaar zijn. Vanwege het niet-deterministische karakter van conceptueel ontwerpen zijn dergelijke conventionele computer technieken echter niet geschikt om hierin ondersteuning te bieden. Daarom is in dit project het gebruik van Kunstmatige Intelligente technieken onderzocht.

Als eerste toepassings gebied is gekozen voor het vliegtuig ontwerp, vanwege de complexiteit. Het gereedschap en de methodiek zijn echter ook toepasbaar om het ontwerp te ondersteunen van schepen, gebouwen en industriële installaties.

Het ontwikkelde ontwerp gereedschap en methodiek is gebaseerd op de bekende "ontwerp cyclus", die in onze versie bestaat uit vier stappen:

1. Het *voorstellen* van een concept: deze taak wordt ondersteund met zgn. "Case-Based Reasoning" (CBR) (voorbeeld gebaseerde redenatie) technieken. Uit een "case base" (data bank), die is gevuld met data van bestaande voorwerpen, wordt één voorwerp geselecteerd waarvan de prestaties het meest overeenkomen met de ontwerp specificaties. Dit voorbeeld wordt gebruikt als startpunt en levert de configuratie of topologie van het ontwerp, samen met de initiële parameter instellingen. Het is mogelijk om twee of meerdere voorbeelden te combineren, wat leidt tot een *geadapteerd* voorbeeld.
2. Het *simuleren* van de prestaties van het voorwerp: deze taak wordt ondersteund met "Rule-Based Reasoning" (RBR) (regel gebaseerde redenatie) en Geometrische Modellering (GM) technieken. De regels in RBR representeren de domein kennis in een expliciet, algebraïsch formaat. Er wordt een netwerk van deze regels gebouwd dat de schaal parameters in relatie brengt met de prestatie parameters. GM wordt gebruikt om een geometrisch model te maken en om met de geometrische relaties om te kunnen gaan.
3. Het *evalueren* van deze prestaties: de "solvers" (oplos algoritmen) van de RBR en de GM technieken maken het mogelijk om snel en makkelijk de parameter waarden te veranderen via het RBR netwerk van relaties en de GM relaties.
4. Het *voorstellen van modificaties*: het ontwerp gereedschap ondersteunt deze taak niet expliciet. Maar, zoals gezegd bij stap 1, een eerste modificatie van de topologie van het voorbeeld (de *adaptatie*) wordt wel ondersteund met CBR. Daarna wordt de cyclus herhaald totdat het concept ontwerp naar bevrediging is ontwikkeld en gebruikt kan worden als invoer voor verdere uitwerking en evaluatie met bestaande CAD/CAE gereedschappen.

Ieder van deze drie technieken, CBR, RBR en GM, is geïmplementeerd in een aparte module. Bestaande programma's zijn gebruikt en aangepast voor deze studie.

Om de opzet en doelmatigheid van het ontwerp gereedschap te testen, is een voorbeeld studie uitgevoerd op het conceptuele ontwerp van een commercieel passagiers vliegtuig voor 80 personen. De studie heeft uitgewezen dat de ontwerp benadering bruikbaar is voor het ontwerpen met geparametriseerde componenten, het zgn. "configuratie-ontwerp" type ontwerpen, maar minder bruikbaar voor het "innovatieve" en "creatieve" type ontwerpen.

Hoewel is gebleken dat iedere techniek geschikt is voor haar taak, zijn er ook beperkingen. Zo vereist iedere techniek een behoorlijke inspanning in de voorbereidende fase, bij het vullen van de data of kennis banken. En er is ervaring nodig om de CBR en RBR modules goed te kunnen bedienen.

Vervolg studies dienen zich vooral te richten op de *adaptatie* van het voorbeeld en de integratie van de drie toegepaste technieken.

Preface

This thesis is the result of the AIDA (Artificial Intelligence supported conceptual Design of Aircraft) project. The aim of this project was to develop and implement a design tool for the conceptual design phase. Existing CAD/CAE (Computer-Aided Design and Computer-Aided Engineering) tools, such as ADAS (Aircraft Design and Analysis System), are more tailored to optimizing existing design configurations and do not provide support for the initial design phase when new or modified design concepts have to be proposed. The objective of the AIDA project was to investigate whether AI techniques could play a role in this phase, and to demonstrate this by actually performing a design case with the aid of AI techniques.

This research started off at the end of 1994 at Delft University of Technology (DUT) as a so-called "Beek" project, sponsored by the university to stimulate innovative multi-disciplinary research. The project resulted in a close cooperation of three sections: the Aircraft Design section of the Faculty of Aerospace Engineering, the Knowledge Based Systems section of the Faculty of Computer Science and the Computer Graphics & CAD/CAM section of the same faculty. The project ended in 2000 at which time I found a job in the industry. Due to these circumstances it took another 4 years to finish the thesis.

October 2004, Eindhoven.

Date Rentema

Contents

Summary v

Samenvatting (summary in Dutch) vii

Preface ix

Contents xi

1. Introduction 1
   - 1.1 Aircraft design 1
   - 1.2 Design 2
   - 1.3 AI and design 4
   - 1.4 The AIDA concept 6
   - 1.5 Overview of the thesis 7

2. The conceptual design of aircraft 9
   - 2.1 Introduction 9
   - 2.2 The aircraft concept 10
   - 2.3 Analysis of the conceptual design phase 16
     - 2.3.1 The design process 16
     - 2.3.2 The domain knowledge 18
   - 2.4 Computer support tools in aircraft design 20
   - 2.5 Conclusions for the proposed computer support system AIDA 29

3. AI and design 31
   - 3.1 Introduction 31
   - 3.2 Case-Based Reasoning 35
     - 3.2.1 Introduction 35
     - 3.2.2 The CBR cycle 36
     - 3.2.3 Case indexing 39
     - 3.2.4 Case matching 39
     - 3.2.5 Case selection 40
     - 3.2.6 Case adaptation 40
     - 3.2.7 Case contents 42
     - 3.2.8 Case representation 44
     - 3.2.9 Reflections upon CBR and design 45
   - 3.3 Rule-Based Reasoning 47
     - 3.3.1 Introduction 47
     - 3.3.2 The RBR components 49
     - 3.3.3 Knowledge representation 51

3.3.4 Inference engine 51
3.3.5 Tha *action* task: solving the relations and constraints 53
3.3.6 The role of RBR in design 54
3.4 Geometric representation 55
3.4.1 Introduction 55
3.4.2 Methods to specify geometry 56
3.4.3 Parametrised geometry 59
3.4.4 Constraint-based geometry 60
3.4.5 Discussion 61

4. Concept of the AIDA system 63
4.1 Introduction 63
4.2 General set-up 65
4.3 Case-based reasoning module 67
4.3.1 Tasks 67
4.3.2 Implementation of case-based reasoning technique 67
4.3.3 Case representation 72
4.3.4 Adaptation with the CBR-module 73
4.4 Functional module 76
4.4.1 Tasks 76
4.4.2 QUAESTOR: the implementation 76
4.4.3 Working with QUAESTOR 79
4.4.4 Implementation aspects of QUAESTOR 80
4.5 Geometrical module 81
4.5.1 Tasks 81
4.5.2 Implementation in Pro/ENGINEER 82
4.6 Central User Interface module 84
4.6.1 Tasks 84
4.6.2 Implementation in C-language 84

5. Design of an aircraft concept with AIDA 87
5.1 Introduction 87
5.2 Design specifications 87
5.3 Initial estimations with the CBR-module 88
5.3.1 Introduction 88
5.3.2 The pre-processing work 89
5.3.3 The target-file 95
5.3.4 Output of the CBR-module 97
5.3.5 Selecting the best case 99
5.3.6 Remarks 105
5.4 Adaptation with the CBR-module 108
5.4.1 Inroduction 108
5.4.2 The adaptation target-file 108

5.4.3 Output of the CBR-module: the similarity scores 111
5.4.4 Selecting the adaptations 112
5.4.5 Generating the adapted case 113
5.4.6 Discussion on the use of the CBR-module 115
5.5 First modifications with the Functional module 117
5.5.1 Introduction 117
5.5.2 The pre-processing work: building the knowledge base 118
5.5.3 Building the templates 123
5.5.4 Performing calculations with the template 130
5.5.5 Combining the results of several templates 132
5.5.6 Evaluation of the Functional module 135
5.6 Visualization of the aircraft concept with the Geometrical module 138
5.6.1 Introduction 138
5.6.2 The pre-defined components 139
5.6.3 Topological constraints 141
5.6.4 The input 143
5.6.5 The output 144
5.6.6 Evaluation 145
5.7 Evaluation of the case study 147

6. Conclusions 149
6.1 Evaluation of the AIDA system and its modules 149
6.1.1 The AIDA set-up 149
6.1.2 The CBR-module 150
6.1.3 The Functional module 152
6.1.4 The Geometrical module 152
6.2 AI in the design process 153
6.2.1 The CBR techniques 153
6.2.2 The RBR techniques 155
6.2.3 The Geometrical modelling techniques 156
6.3 General recommendations 156

References 159

Acknowledgements 165

Curriculum Vitae 167

1 Introduction

This thesis describes the set-up of a computer support tool for the conceptual design of aircraft using Artificial Intelligence (AI) techniques. This subject implies the involvement of several topics: the conceptual design process, the design process of aircraft and the supporting role of AI techniques. Based on these topics this chapter gives an introduction to the project called AIDA: Artificial Intelligence supported Design of Aircraft. At the end of the chapter the objectives of AIDA are outlined and an overview of the thesis is given.

1.1 Aircraft design

Aircraft design is a complex problem, comparable with the design of other complex objects and systems like buildings, industrial appliances, and ships. The requirements list for an aircraft design may contain a large number of issues, from functional requirements such as the aircraft's size, capacity, payload-range combinations, to technical requirements such as flight control, aerodynamics, stability of its structures, production, costs, environmental impact, etc.. After analysing these requirements, the designer may come up with an initial conceptual solution. In case of aircraft, this first concept will describe the configuration of the aircraft in global terms, i.e. its general arrangement and its size in terms of weight, thrust or power, wing span and area, etc. At this point it is difficult to assess the proposed solution. Existing numerical and analytical tools for aircraft design, such as ACSYNT [Mason, 1993], AAA [Roskam, 1989], RDS [Raymer, 1996] and ADAS [Bil, 1988] [Bil, 1989], require much more detail than is available in the conceptual design phase. However, it is very expensive and time-consuming to elaborate the initial design into a fully elaborated product, only to learn that the proposed solution in the end does not satisfy the functional requirements.

Another drawback of existing CAD tools is that they do not support the suggestion of initial configurations. Therefore the designer has to rely on his experience, and on examples of existing aircraft. Further some simplified physics, statistical data and empirical rules may be available. The designer may have deduced some 'rules-of-thumb' from these physics and experiences, which results in lists of advantages and disadvantages coupled to each configuration choice. These lists are dominated by qualitative arguments which makes it difficult to draw conclusions.

Against these issues, which make the conceptual design of aircraft complicated, stands the convenient feature that most complex objects are composed of a set of basic components. In case of aircraft: the fuselage, the wing, the tailsurfaces, the undercarriage and the powerplant. Because conceptual design only deals with approximate sizes, such a decomposition of the aircraft greatly enhances the overview of the design possibilities. For conventional aircraft the solution space is then reduced to a limited number of configuration alternatives: high-, mid- or low-wing; T-tail, cross-tail or conventional tail; turbofan, turboprop or piston engines; wing- or fuselage-mounted engines; etc. However, the number of possible configuration combinations is still too large to be fully elaborated.

In this thesis a new design methodology for conceptual design is examined, using several AI techniques. Together these techniques support the total design cycle of proposing and evaluating solutions.

Before introducing these AI techniques some more words are spent on the characteristics of designing.

1.2 Design

Design is an activity that generates a 'materialised' solution for a 'functional' problem, i.e. it proposes a system or 'structure' that shows some 'behaviour' that satisfies the 'functional' demands. However, there is no direct reasoning possible from functions to behaviour and structure, i.e. a logical, straightforward deduction of the structure from the function is not possible. It is even feasible that more than one structure is capable of realising the demanded functions, or none at all. Hence, designing is a non-deterministic process.

Designing has more challenging features. Typically, it is often an 'ill-defined' process, meaning that the functional requirements may change during the process, for example when they appear to be too conflicting. Also time and resources may be limited, and we may have to decide in a situation of uncertainty. Another challenge is to define the optimality criteria based on the diversity of design specifications, in order to generate an optimal design or select the best feasible design.

Because straightforward reasoning is not feasible, a 'generate-and-test' strategy is often applied. This strategy leads to a number of iterations through the design cycle which is visualized in Figure 1.1.

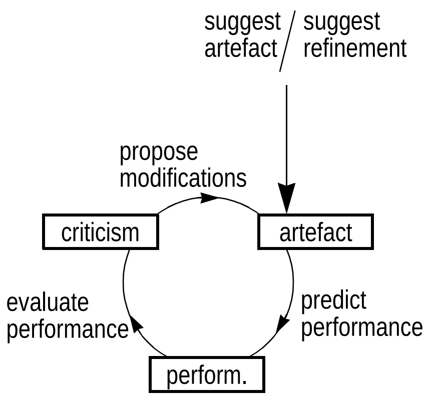


Figure 1.1. The basic tasks in the design cycle.

Five tasks are distinguished. The design procedure begins with the *suggestion* of an artefact, i.e. the materialised solution, supposing that the specifications have been defined. The designer usually has to rely on his or others experience and insight for this task, since proper methods are lacking.

Then the performances or function of the suggested artefact are *predicted*. The designer already has an idea about the performances and other properties when suggesting the artefact, but only when the artefact has been defined, its characteristics can be properly predicted. In general good prediction methods are only available for more detailed artefacts. The term predicted is used instead of analysed to emphasize the limitations about the validity and accuracy of the methods due to simplifications.[1]

Next the predicted performances are *evaluated*. With the accuracy of the predictions in mind, the predicted performances and other properties are compared with the required ones. The designer judges the shortcomings and marks the properties which need further attention.

Based on the criticism *modifications* are proposed. The purpose of the modifications is to improve the artefact so that it will better meet the requested performances and other properties. Another possibility is that the initial requirements are modified when they are regarded to be too restrictive.

When the artefact is evaluated to be good enough, the designer *suggest refinements*. This means that the artefact will be described more completely by adding details, for example by focusing on a component of the artefact. This also involves the refinement of the requirements about the performances and other properties, which are based on the design decisions which have been made previously.

1. In case of aircraft design the analysis phase starts not until the hardware is available (flight tests, structural tests, etc.).

An often applied strategy to avoid exhaustive generate-and-test search is the 'divide-and-conquer' method. The problem is decomposed into several sub-problems that are easier to solve and the different sub-solutions are then combined into one integrated solution. An example of such a decomposition strategy is the 'function-means' method. Requirements are recursively decomposed until a solution can be provided and the part-solutions are then successively combined. This decomposition strategy does not solve the design problem itself but transposes it to the 'integration' activity; the initial decomposition and analysis may provide some useful clues for the integration phase, however. The decomposition-integration cycle can be repeated for different levels of detail, or different functional decompositions.

The integration step can be avoided when an existing solution is taken as a starting point. The chosen solution, which partly satisfies the new functional requirements, is then refined via a 'reuse-and-adapt' or 'diagnosis-and-repair' cycle. The solution is analysed and modifications are proposed that relieve some of its weaknesses.

This approach is better suited for so-called 'routine design' problems than for 'creative' designs where new principles are applied in a new context. Most designs fall in between routine design and creative design: a new design is often a new composition of existing components. According to [Mittal, 1989] this type of design can be classified as 'configuration design'. Typically large-scale design problems such as design of aircraft, buildings, ships and industrial appliances can be characterized as configuration design: certain aspects of a new design may be 'innovative' but in most cases there will be quite some similarity with earlier designs.

Although the reuse of existing components greatly benefits the design process, the number of possible combinations is still far too large to be fully elaborated. Hence a good initial choice is very important.

In the next section we will review to what extent AI techniques can be used to cope with these issues.

1.3 AI and design

Within the area of designing, Artificial Intelligence (AI) is concerned with the application of knowledge. Within AI, three main directions of reasoning can be distinguished: reasoning by logic, reasoning by learning, and reasoning by analogy. Expert systems apply rule-based reasoning (RBR) technique, a form of reasoning by logic. An expert system is useful when the domain knowledge can be formalised into simple rules, such as mathematical problems, and when common sense does not play an important role. Unfortunately, design can not be formalized this way. Reasoning by learning can be implemented with Artificial Neural Networks (ANN). An ANN consists of a network of nodes (processing elements) connected via adjustable weights (connections). By training the network with a large set of input-output pairs, the system learns the functional relation between the input and the output space. This type of generalized learning can not be applied to the design problem when the solution space is sparse and discontinuous. The third form of reasoning, reasoning by analogy, is best exemplified by Case-Based Reasoning (CBR). Cases are stored in a case-base to create a reservoir of problem-solution combinations. When a new problem is presented, CBR searches for cases with similar problem descriptions. Although the retrieved case usually does not completely fit the new problem, the retrieved solution may be a good starting point for further adaptation and optimization. The difference with the other AI methodologies is that CBR does not use generalized domain knowledge but knowledge which is locally valid: the implicit knowledge within a case which relates a problem with a solution only holds for that particular case. See Figure 1.2.

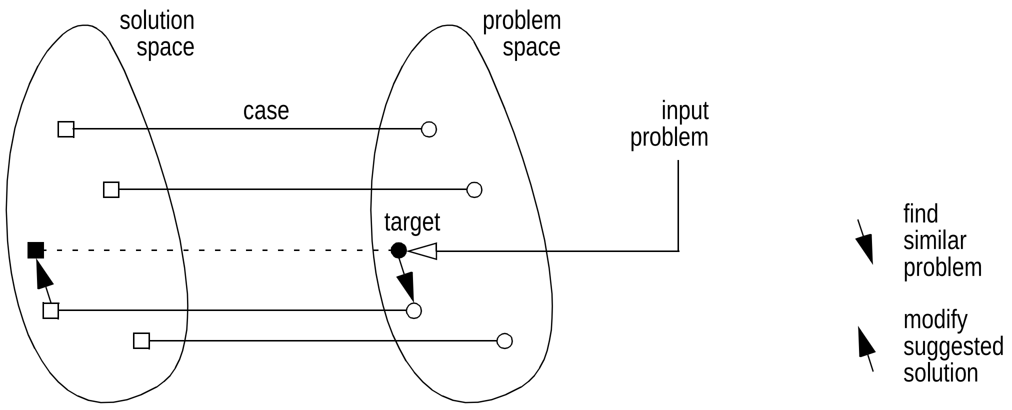


Figure 1.2. Principles of problem-solving with Case-Based Reasoning (from [Leake, 1996]).

Considering the characteristics of conceptual design, reasoning by analogy seems to be a helpful method to assist the suggestion of a concept. Because CBR applies implicit knowledge, this paradigm offers interesting possibilities to overcome the lack of (quantitative) knowledge at this design phase. CBR in design, however, differs from other CBR applications where the number of cases is generally quite large but each individual case is relatively simple. In design the examples are few but often complex and may cover a large diversity of the functions. Hence, to be able to use CBR properly, the information about the design cases will have to be encoded in a flexible way.

A useful representation scheme for design cases is with the triad Function-Behaviour-Structure [Gero, 1992]. Structure represents the materialized version of the design: the geometry, the components and the assembly. Function represents the performances of the functional aspects of the model. Behaviour describes more explicitly how the structure is able to accomplish the functions; i.e. the mechanism by which the (functional) problem is solved. The behaviour component is very much dependent on the structure. Given two different design configurations, there may be a completely different set of behaviour data to evaluate each of them.

In addition to functional requirements, the list of specifications may also contain requirements of the behaviour and structure of the design. Some functionality may therefore be an indirect result of behaviour and structural choices in the list of requirements, and not directly described by the list of requirements.

For a proper evaluation of the structure via the behaviour and the function components, explicit knowledge can be applied. Since its logical character, the RBR technique seems suitable for the implementation of this knowledge. Compared with the conventional techniques implemented in existing computer support tools, this technique may improve flexible use of domain knowledge. By logical reasoning, the input and output of the calculation methods can easily be matched in order to determine a proper evaluation sequence.

Also the transparency of the applied knowledge will be improved. Instead of performing complicated calculations within a module, RBR can be used to split up these calculations into basic functions and connect them on-line by logical reasoning.

Strongly associated with the evaluation functions are constraints and relations imposed on the (standard) components of the structure. These constraints keep the structure functionally consistent, encode some form of parametrization of the structure, and/or be part of the interface specification between components. These constraints are not directly used to evaluate the design but to model the design and put some intelligence into the configuration and adaptation process.

1.4 The AIDA concept

Within the AIDA project several AI techniques are examined to support the conceptual design of such complex objects as aircraft. It is supposed that these techniques can cope with the non-deterministic character of the design task, where conventional computer techniques are not suitable.

A system set-up is presented consisting of three main modules:

- a Case-Based Reasoning (CBR) module:
  to propose an aircraft concept;
- a Functional module:
  which implements Rule-Based Reasoning (RBR) techniques for the functional evaluation of the aircraft concept; with the RBR techniques relations can be created between the structure, behaviour and functional components in a flexible and transparent manner;
- a Geometrical module:
  to implement geometrical constraints within and between the aircraft components.

For each module existing tools have been used, operating on different platforms. This will obstruct the communication between the modules, but that is taken for granted because the emphasis of this study is on the exploration of these distinct techniques. The advantages and draw-backs are described, the practical implications, etc. Hardly any attention is given to the integrated implementation of the separate techniques.

1.5 Overview of the thesis

For a better understanding of the problem a typical aircraft conceptual design process is described first. Chapter 2 gives a description of the process and what is understood by an aircraft concept. Further some typical design issues are mentioned. Also current computer support tools are examined which have been developed for aircraft design. This leads to some deliverables and directions for the AIDA project.

Chapter 3 deliberates on the means of suitable design support techniques. The Case-Based Reasoning method is examined and its typical issues are reviewed when applied in design problems. An important issue is a flexible case representation. Also the use of basic aircraft components and their configuration and geometric constraints is described. The Rule-Based Reasoning technique is studied to support the evaluation task in the design process, i.e. the use of functional constraints.

The next chapter describes how these techniques are applied for this particular goal: to support the aircraft conceptual design process. A system set-up is presented consisting of three main modules: a CBR-module to propose an aircraft concept, a Functional module which implements RBR-techniques for the functional evaluation of the aircraft concept, and a Geometrical module for the geometrical evaluation. For each module existing tools have been used.

In Chapter 5 a case study is carried out to reveal the typical aspects of the various techniques. Based on a set of specifications a case is selected from the case-base, which is adapted by combining parts of another case. Sensitivity studies of some primary design parameters are performed by the Functional module. The resulting aircraft concept is modelled in the Geometrical module.

The advantages and disadvantages of the different techniques which have been explored in this case study are collected and discussed in the last chapter.

2 The conceptual design of aircraft

2.1 Introduction

The aircraft design process is a long and complicated process. Due to the complexity of the systems and the high demands on the aircraft, a large number of highly qualified people is involved. Usually it will take 5 to 10 years to develop an aircraft from scratch to the first production.

Within the aircraft design process several phases can be distinguished [Torenbeek, 1982]. The phases are shown in Figure 2.1.

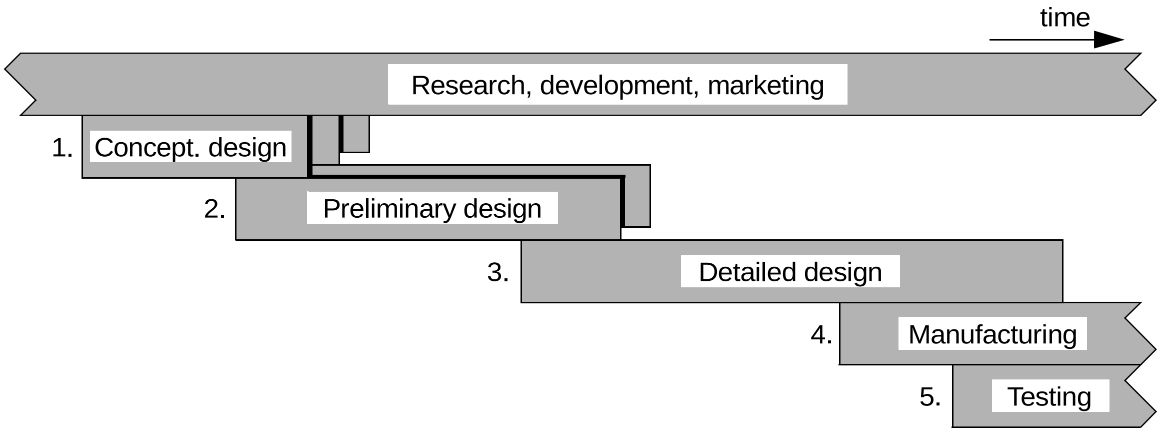


Figure 2.1.   The several phases in aircraft design (from [Torenbeek, 1982]).

The design of an aircraft is initiated by the marketing and/or the research and development departments. The marketing department continuously studies the airliner business. They try to reveal the market trends in order to predict the airliner needs for the next 20 years. The research and development people are trying to increase the qualities of the design methods, materials, production and maintenance techniques, software tools, etc., which can be used to design better aircraft at lower costs and in less time.

The marketing department draws up the design specifications which are used for the conceptual design, i.e. the first phase of the design process. In this phase one or more feasible aircraft concepts are created by the department in charge, which includes the overall architecture i.e. the configuration and the main dimensions. The configuration describes the topology of the aircraft, for example the type of tail, the type and number of engines and where they are located, etc.. The main dimensions roughly describe the shapes and sizes of wing, tail and fuselage, and include some other parameters such as the weight and the thrust or power.

The level of detail in this design phase is low. The design problem of the complete aircraft can usually be decomposed into design problems of the standard parts of an aircraft: the fuselage, wing, tail surfaces, engines and undercarriage. In the conceptual design phase the interaction between the sub-problems is high, so the sub-design problems are solved simultaneously.

In the preliminary design phase the decomposition of the design problem continues. For example wing components such as the flaps and ailerons are considered. The level of detail increases, but the interaction between the sub-problems decreases; although modern techniques and methods such as system engineering and concurrent engineering are employed to keep the interaction as closely as possible. Subsequently more departments may be involved. This tendency continues in the detailed design phase, when specialized departments work out the preliminary design.

The design process is finished when the aircraft is manufactured and the tests have been successfully concluded.

2.2 The aircraft concept

The goal of the conceptual design phase is to generate one or more aircraft concepts which will meet the design specifications. The specifications come from the customers requirements and the airworthiness requirements. They are formulated into quantitative constraints to allow precise and unambiguous evaluations.

Typical customers requirements are the number of passengers in combination with the range, the cruising speed, the take-off and landing distances, etc.. The airworthiness regulations define for example minimum aircraft performances with one-engine inoperative, such as the minimum climbing gradient and stall characteristics. Beside of these performances also requirements are specified about the costs, maintenance, etc..

An aircraft concept includes the aircraft configuration and its primary dimensions. The configuration describes how the aircraft components are located with respect to each other, and how the components themselves are defined, i.e. the type of components. For example a wing-mounted engine configuration differs from a configuration with engines mounted on the fuselage; and the definition of a turbofan powered aircraft is different from the definition of a turboprop powered aircraft, involving other design parameters.

The primary dimensions specify for example the shape and size of the wing, the tail and the fuselage. Other sizing and behaviour parameters are included, such as the maximum take-off weight, the maximum take-off thrust or power and aerodynamic coefficients.

The procedure to derive an aircraft concept from the design specifications is not formalised. Initially the design space is very large and explicit relations between the input and output are sparse, especially between the performances and the configuration. Therefore many arguments applied in this design phase have a qualitative character. It takes an experienced designer to evaluate them properly, and comparable aircraft are often studied.

The explicit knowledge available is based on heuristics, simplified physics and statistics, and has a limited accuracy. Examples of these numerical methods have been collected in educational books such as [Raymer, 1992], [Roskam, 1987-1990] and [Torenbeek, 1982], in the ESDU series [ESDU], and have been implemented in design tools such as CAPDA [Haberland, 1984], ACSYNT [Mason, 1993], AAA [Roskam, 1989] and RDS [Raymer, 1996].

In the design process some issues can be distinguished which usually show up during most conceptual design phases:

- the size and shape of the fuselage;
- the design weights, such as the maximum take-off weight (MTOW);
- the lift curves and drag polars;
- the size and shape of the wing and its location with respect to the fuselage;
- the wing loading vs. thrust loading diagram;
- the size and shape of the tailplanes;
- the powerplant and its location;
- the load and balance diagram.

The size and shape of the fuselage

The lay-out of the cabin is based on the specified number of passengers, seat pitches, aisle widths and passenger seating configuration. Also other items such as the cargo compartments and aircraft systems have to fit inside the fuselage. Because current commercial airplanes fly at high altitudes, a pressure cabin is required. From the structural viewpoint the optimum shape of the fuselages cross section is therefore circular. Hence circular cross sections are commonly applied, but also double-bubbles or oval shapes. Suitable diameter length ratios of the nose and the tail of the fuselage are based on the examination of comparable aircraft.

The lay-out of the cabin and the location of the aircraft systems determine the location of the fuselage’s centre of gravity, and therefore affect the location of the other aircraft components. This is expressed in the load-and-balance diagram; see Figure 2.3.

The design weights

The Maximum Take-Off Weight (MTOW) is a leading parameter in the overall design process. Therefore an initial estimation is performed as soon as possible in the design process. This also holds for the Operational Empty Weight (OEW), which is the aircraft weight without fuel and without payload, the Maximum Zero Fuel Weight (MZFW), which is the maximum weight without fuel, and the fuel weight with its volume which are important for the determination of the range. Several methods are available: the more details are known, the more accurate the method. Often comparable aircraft are examined to estimate realistic values. Other methods involve weight estimations of the aircraft components, often expressed by non-dimensional weight ratios.

The lift curves and drag polars

The lift-curve shows how the lift coefficient c_L depends on the angle-of-attack α and which maximum c_L-value the (clean) wing can reach. This is an indication of how the aerodynamic behaviour changes with the angle-of-attack of the aircraft and what kind of high-lift devices are required to be able to reach a requested c_L-value during take-off and landing. The lift curve is considerably influenced by the shape of the airfoil(s) and the high-lift devices such as the flaps and slats.

The drag-polar represents the relation between the lift and the drag (coefficients) of the aircraft. It defines the aerodynamic fineness ratio L/D (lift over drag), which is important for example for the required thrust or power with respect to the aircraft weight. The polar is usually expressed by the basic equation:

$$c_D = c_{D_0} + c_{D_i} = c_{D_0} + \frac{c_L^2}{\pi A e} \qquad \text{(Eq. 2.1)}$$

There are simple methods to estimate the involved parameters. For example the determination of the Oswald factor e, which expresses the aerodynamic efficiency of the wing, can be based on statistics. The aspect ratio A is often subject of sensitivity studies, but initial values can be based on the examination of comparable aircraft. Several methods are available to estimate the zero-lift drag coefficient c_{D0}. Again statistics can be used, or the problem can be translated into the estimation of the contributions of the aircraft components, which depend on the external shape of the aircraft. However more thorough methods are preferred for their better accuracy and sensitivity to variations in the geometry.

The size and shape of the wing and its location

The main function of the wing is to carry the aircraft; i.e. to generate enough lift to get and keep the aircraft off the ground. The shape of the wing influences the drag-polar, i.e. the lift-drag ratio, whereas the size determines their absolute values. Also the structural weight and the volume of the fuel tanks, usually buried in the wing, are important factors.

Some wing parameters are explicitly involved in numerical methods, such as the wing area S and the aspect ratio A (see Equation 2.1). Therefore they can be subjected to sensitivity studies to estimate proper values. The influence of other parameters are harder to be quantified at this stage, such as the wing sweep Λ and the wing thickness ratios t/c. At this design phase the estimation of their values is usually based on some statistics and on the knowledge of the designer about their affects; for transonic aircraft the sweep and thickness ratios of the wing for example influences the normal and maximum cruise Mach number.

The (longitudinal) location of the wing with respect to the fuselage is governed by stability and control requirements on the aircraft. The stability demands that for conventional aircraft the aircraft’s centre of gravity (c.g.) is located in front of the neutral point (n.p.) or aircraft aerodynamic centre (a.c.); the a.c. is the point where the pitching moment is essentially constant at different angles of attack, i.e. a suitable point to consider the resulting aerodynamic force variations to be acting from. When the distance is too big, however, the forces necessary to control the aircraft become too large. For this complicated issue the load-and-balance diagram is used; see Figure 2.3.

The vertical location of the wing is greatly influenced by the powerplant location: for wing-mounted engines the ground clearing may become critical, and fuselage-mounted engines should not be located in the wake of the wing. Also the effects on the structure of the fuselage and some aerodynamic characteristics are considered. However, it is difficult to quantify these effects at the conceptual design phase.

The powerplant and its location

The engines must balance or out-balance the aircraft drag during the different flight-phases such as the take-off, climb, cruise, descend and landing, with all engines running and in one-engine-inoperative situations. The maximum thrust or power has a large influence on the aircraft performances and costs. The required thrust or power can be estimated with numerical methods concerning these performances. The most simple methods require data about the shape and sizes of the aircraft and the aerodynamic coefficients, i.e the primary dimensions. The results of these methods can be presented together in thrust-loading-wing-loading diagrams; see Figure 2.2.

The number of engines affects the possible configuration of the aircraft and the performances in one-engine inoperative conditions. These considerations and the required thrust or power in combination with the availability of the engines generally governs the number of engines to be applied.

The type of engine, for example turboprop or turbofan, determines which parameters to study: the power or the thrust, respectively. Each type of engine has a different thrust or power lapse characteristic and fuel consumption. It depends on the type of flight missions, in particular the maximum flight speed, which type of engine is preferred.

Because the weight contribution of the engines is large, their configuration influences the location of the other components for a suitable location of the centre of gravity; see Figure 2.3. Other arguments which direct the choice of the engine configuration are primarily qualitative, such as accessibility of the engines concerning the maintenance, influence on the wing configuration, general aircraft handling characteristics, etc..

The wing loading vs. thrust loading diagram

The loading diagram displays the relations between the thrust loading T/W and the wing loading W/S; an example is shown in Figure 2.2.

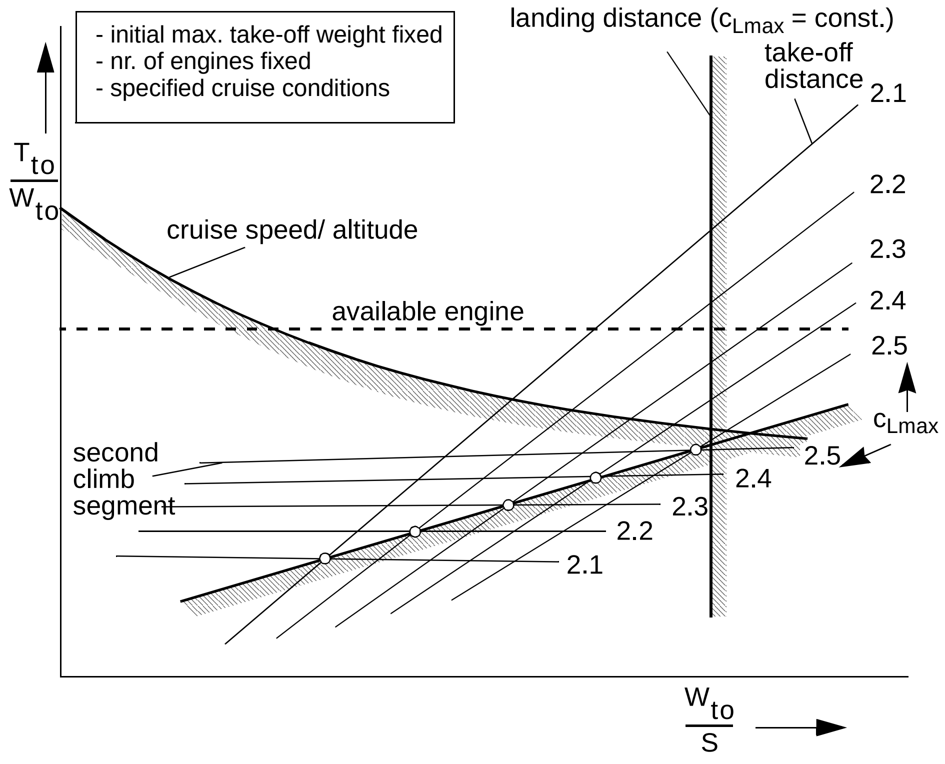


Figure 2.2. A wing loading vs. thrust loading diagram.

The diagram is used to choose proper T/W-W/S combinations with respect to performance requirements for which numerical relations between T/W and W/S are available. The diagram allows different design specifications to be considered simultaneously. For given MTOW the T/W- and W/S-values are applied to determine proper values of the maximum take-off thrust (or power) and the wing area S.

Each line in the diagram represents the T/W-W/S-relation for a specific performance requirement. Together these lines form the boundary of the design space with respect to suitable T/W-W/S-combinations. Other lines may be created which indicate optimum combinations regarding other design parameters.

The numerical relations are based on empirical relations, statistical data and (simplified) flight mechanics. They involve many parameters about the aircraft configuration, fuselage, weight, drag-polar, wing and powerplant and other primary dimensions. In some relations premature estimations are required about the weight and the thrust or power. This creates a loop in the calculations: the weight is determined by the wing- and thrust-loading, which again depend on the weight. Initial weight estimations are needed to be able to start these calculations.

The size and shape of the tailplanes

The primary function of the tailplanes is to control the stability and the manoeuvrability of the aircraft. The horizontal tail delivers the aerodynamic forces required for the longitudinal stability and control of the aircraft. The function of the vertical tail is to provide the lateral control and stability about the vertical axis. Since the tailplane loads are directly related to their surface areas, their sizes are directed by the distances of the tailplanes to the aircraft's c.g..

The type of tail, i.e. the location of the horizontal tailplane with respect to the vertical plane, gets more attention. Its location is strongly influenced by the wing wake, the propeller slipstream or the jet efflux. To stay effective the tailplane should remain outside these areas. These arguments are based on geometrical considerations.

The load and balance diagram

The load and balance diagram presents the location of the aircraft's centre of gravity (c.g.) for different loading conditions. This location is usually expressed relative to the Mean Aerodynamic Chord (MAC). The MAC is a characteristic chordlength used to represent the overall wing pitching moment.

The loading conditions change due to shifting distributions of the payload and the fuel. The c.g. should remain between boundaries for a stable and controllable aircraft. Figure 2.3 shows a typical load and balance diagram.

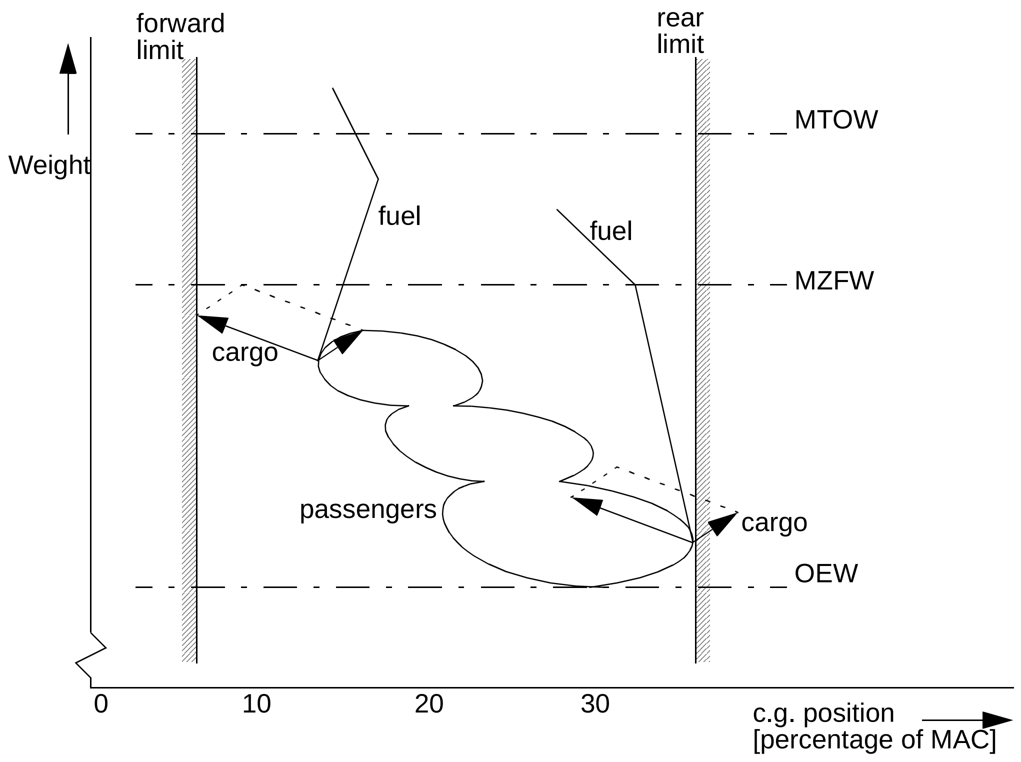


Figure 2.3. A load and balance diagram.

The c.g.-lines show the c.g.-shift due to different numbers of passengers, filling the cargo compartments (forward and rear) and different fuel volumes in several fuel tanks. The horizontal lines show the design weights: Maximum Take-Off Weight (MTOW), Maximum Zero-Fuel Weight (MZFW) and Operational Empty Weight (OEW). The forward and rear limits are determined by the stability and control considerations, such as longitudinal static and dynamic stability, trim and elevator control forces.

To be able to generate such a diagram, a thorough examination of the different components should be performed. Especially the arrangement of the fuselage is important, with the cabin lay-out and the distribution of many aircraft systems.

2.3 Analysis of the conceptual design phase

2.3.1 The design process

As has been mentioned before, a fixed design path for the design of aircraft does not exist. Some issues have been distinguished, though, which play a role in almost every conceptual design process of aircraft. These issues are analysed for a better understanding of the design activities involved in the process. For that purpose the design cycle as explained in Section 1.1 is used. Emphasize is laid upon the tasks mentioned in Figure 1.1: *suggesting*, *predicting*, *evaluating*, *suggesting modifications* and *adding details*.

The load-and-balance and the wing-loading-thrust-loading diagrams, Figure 2.3 and Figure 2.2, are both *evaluation* tools in the conceptual design phase. The wing loading vs. thrust loading diagram presents the results of several (numerical) *prediction* methods. With these methods some of the specified (quantitative) performance parameters are related with some design parameters, i.e. the primary dimensions. Each line in the diagram expresses the interaction between several primary dimensions for a particular specification. Based on the diagram a suitable combination of W/S and T/W is *suggested*, together with proper values for c_{Lmax}, A and some other primary parameters.

The *suggested* values for other primary parameters, such as the wing sweep Λ and thickness ratio t/c, have hardly any influence on the diagram. Hence the diagram can not be applied to *evaluate* these parameters. Proper *evaluations* are only later possible, using *prediction* methods which require more details. This also holds for most of the configuration *suggestions*, which hardly affect the diagram.

For the parameters for which quantitative *prediction* methods are missing or not usable, the designer tries to come to qualitative *predictions*. He uses his and others experience, statistics and rules of thumb, and studies comparable aircraft in trying to *predict* the consequences of *suggested* values. The *evaluation* of these *suggestions* is difficult due to the inaccuracies of the *predictions* and their limited validity.

The load and balance diagram shows the effects of the weight *prediction* of the components and their *suggested* locations. The *prediction* of the weights is based on heuristic relations and statistics, combined with the experience of the designer. For proper *predictions* a certain level of detail of the involved components is needed. Especially the wing has to be defined, since its MAC is needed for the diagram. With regard to the locations the fuselage lay-out is of particular importance, including all its aircraft systems. To be able to *suggest* the locations a geometrical model of the components is very convenient. For example the size of the horizontal tail determines the c.g. limits to a large extent.

The *suggestion* of the outer shape and size of the fuselage seems to be a merely isolated problem at this design phase. However, the lay-out of the cabin and the aircraft systems inside the fuselage has a large impact on the load-and-balance diagram, and therefore on the locations of for example the wing and the engines. The *evaluation* of the *suggested* lay-out inside the fuselage is thus possible by generating that diagram.

The drag polar relation, Equation 2.1, is applied in a number of numerical prediction methods necessary for the wing-loading-thrust-loading diagram. In this perspective it is part of the *prediction* methods.

A proper value for c_{D0} can directly be *suggested* by the designer, based on his experience or statistical tables which have been collected in for example [Torenbeek, 1982]. The designer can also apply relations which estimate the contributions to c_{D0} of the aircraft components. These relations are part of the *prediction* methods too.

The W/S- and T/W-values in combination with the suggested initial values of the maximum weight or thrust, define the values for the wing area, the maximum weight and thrust. Usually the maximum thrust will be adjusted to the available thrust of existing engines, and the weight and wing area are revised accordingly. This is the suggest modifications task in Figure 1.1. During the process when more details are added, the prediction methods will become more accurate and reliable, so the weight will continuously be modified. The modifications should be kept as small as possible, though, to keep the suggestions about other parameters which are based on the weight valid.

All *suggested* values serve as guidelines for the next design steps. That is, they are *added* to the design specifications. They increase the level of detail of the designed artefact.

For example a proper value of the maximum liftcoefficient at take-off is based on the *evaluation* of the loading diagram. In next design steps the wing will be designed for that value.

2.3.2 The domain knowledge

The previous section described how the design tasks could be analysed; it dealt with the roles of the domain knowledge, how the domain knowledge is applied. This section concentrates on the domain knowledge itself. The focus is on typical aspects of the conceptual design of commercial aircraft.

An important aspect is the distinction of standard aircraft components. Most aircraft are composed of the same components to fulfil the same (primary) functions:

- the fuselage
  to carry the payload and fuel, to accommodate the pilot and control systems, and to form a platform for the other components;
- the wing
  to create lift, carry fuel, and often to form a platform for the powerplant and the undercarriage
- the powerplant
  to generate thrust;
- the tailplane and control surfaces
  to provide the means for stability and control in all directions;
- the undercarriage
  to carry the aircraft during ground operations;
- the aircraft systems
  to perform supporting tasks for fuel supply, control of the aircraft, comfort of the passengers, etc..

Each standard component is described by a different set of parameters, due to the different functions and different topologies. Per component different types can be distinguished, which also requires different definitions. For example the definition of a turboprop powerplant contains the propeller diameter, number of propeller blades and the power, whereas the turbofan engine is described by the by-pass ratio and the thrust.

The number of component types are limited, considering regular, commercial airplanes. Especially in the first design phase this aspect reduces the design space considerably.

Another aspect which reduces the design space is the limited number of lay-outs of regular, commercial airplanes; i.e the standard components are arranged in a limited number of configurations. For example the engines are either mounted on the (aft) fuselage or on the wing; the wing is located on upperside of the fuselage, the underside or in between; the horizontal control surfaces are mounted on the aft fuselage, on top of the vertical tail or halfway on the vertical tail, etc..

Though this observation of a reduced design space should decrease the complexity of the conceptual design process, the lack of appropriate design knowledge opposes that. The knowledge required to choose proper configurations and component types is very qualitative and often not explicitly available. In terms of Figure 1.1, suitable predictions methods are missing to perform proper evaluations about suggested configuration choices.

The designer uses his experience and basic physical principles to predict possible effects of his configuration choices. For example the effects of wing-mounted engines on the wing design: in ground operations the engine weight increases the downward load of the wing, but in flight the extra weight reduces the upward load caused by the lift. Only by experience the designer can value both arguments and conclude what effect the engine configuration might have on the construction weight of the wing.

The designer also examines comparable aircraft to support the decision process. An existing aircraft shows which performances are possible with a specific configuration. By consulting comparable aircraft the designer uses the design experience which is implicitly accessible in these designs.

These qualitative evaluations are not only applied to choose a configuration, which involve discrete parameters, but also for the initial estimation of continuous parameters, such as the weight and thrust. Especially the comparable aircraft form a good source for initial estimations.

The numerical methods used to create the loading diagram allow some quantitative evaluations. The methods consist of several mathematical relations which are based on simplified physics, empiricism and statistics. Due to the simplifications most relations have a limited validity and accuracy. So for different design specifications different relations are used. The designer should know the restraints well to be able to apply the relations properly.

Due to the interdependencies of the relations, they can not randomly be used. For example the zero-lift drag coefficient has to be estimated before the required thrust can be valued; see the objectives mentioned in Section 2.2. Hence the designer should have an overview of the available relations, in order to apply them in the appropriate order.

So, although the quantitative knowledge helps the designer in *evaluating* the *suggested* concept, a lot of skill is still required to use that knowledge properly.

2.4 Computer support tools in aircraft design

Because of the strong processing power of computers, many computer tools have been developed which exploit this power in order to support the designer. Some tools focus on the graphical modelling of the aircraft, replacing the traditional drawing boards. Advantages are that drawings are easily modified, simply and accurately reproduced, and that the drawing data can be stored and transported without difficulty. It requires a well-organised computer infrastructure, however. These tools help to define the design object, i.e. to model the aircraft geometry. The visualisation of the aircraft is effective for understanding the design, that is for evaluating the artefact in terms of Figure 1.1.

Other tools exploit the computer power to perform numerical calculations. Generally the calculations are part of the *prediction* task of the design process. The first tools were automatic calculators to perform analytical calculations. Then numerical algorithms were developed to solve more complex problems. Because of the limited power of the older computers much effort was put in the development of time- and memory-efficient methods. These methods were intended to solve numerical problems by approximation. Modern techniques have been developed with analytical capabilities using symbolic methods.

Current computer-aided design tools integrate both the modelling and the numerical capabilities. For example Computational Fluid Dynamics (CFD) tools perform numerical calculations on (parts of) the aircraft geometry. Also progress has been made to integrate the *prediction* and *evaluation* tasks and to *suggest modifications* as part of the optimisation cycle.

Further developments within computer-aided design tools aim to improve their accuracy and reliability, and to relieve the designer from his/ her managing tasks. The first aspect is based on the immense rise of computer power and on the developments of more efficient numerical algorithms, which makes it possible to decrease the number of mathematical and physical simplifications. The latter aspect is focused on the flexibility of the tools, and merely deals with the control of the numerical algorithms.

In this section computer tools are evaluated which are used in the first phases in aircraft design. Only those tools are described which deal with the complete aircraft and cover several disciplines, such as aerodynamics and structures. The many sophisticated but highly specialised computer tools, focused on specific areas, are ignored.

CDS: Configuration Development System [Raymer, 1982]

CDS is a conceptual design and analysis program developed by the Rockwell company. The design part consists of so-called "designer’s media", containing a (for that time) advanced geometric modeller to define the shape and size of the aircraft components and their lay-out. Much attention is paid to the capability to create smooth, complex shapes. By storing a set of modelling commands the user can create standard parts with pre-defined design parameters such as wing area and aspect ratio. Also non-geometrical parameters are connected to the components.

In the analysis mode CDS can calculate surface derivatives from the 3-dimensional model. The analysis part acts as an intermediary for commercial, off-line packages, but also contains some on-line analysis tools, such as a wave drag module. Centres of gravity can be derived from the model, assuming uniform weight distribution, or by component related routines. The on-line modules are only used for simple calculations.

[Raymer, 1982] emphasizes the geometry related capabilities of CDS. The paper shows the complex shapes which can be modelled. The re-use of (standard) parts seems to be a convenient way to save modelling time, but this probably requires a good knowledge of the definition parameters because of the complexity of the geometry. Simple sizing routines help the designer with the sizing of the components. For the non-geometrical analyses CDS primary relies on external codes, facilitating the appropriate interfaces. The internal analysis routines seem to depend strongly on the applied geometry definition.

The current status of the system is not clear; from time to time software updates become available.

Paper Airplane [Elias, 1983] [Elias, 1985]

In 1982 the experimental Paper Plane project was started at the Flight Transportation Laboratory of the Massachusetts Institute of Technology (MIT). The aim of the project was to reduce the inflexible, procedural character of existing computer tools. This is caused by the specified set of input and output data of each calculation module, which requires a specific order in which they can be applied. Elias [Elias, 1985] calls it a fixed computational path.

The project concentrated on the use of simple heuristic relations, with the same level of detail as those mentioned in [Torenbeek, 1982] and [Roskam, 1987-1990]. By selecting a set of relations and specifying some parameters, the system determines the computational path. An important feature is the inverted use of the relations. Paper Plane does not discriminate between input and output parameters: when one of these parameters is unknown, Paper Plane can deduce it with the relation using numerical techniques. Because of this feature the designer can implement simple design functions without having to consider its possible use with respect to input and output parameters. The papers do not mention any graphical output or input of the Paper Airplane.

No progress has been reported since about 10 years.

CAPDA: Computer-Aided Preliminary Design of Aircraft [Haberland, 1984] [Xie, 1999] & FLYING OBJECTS [SCHNEEGANS, 1998]

In [Haberland, 1984] the development of a design support tool for commercial aircraft is described, carried out on the Technische Universität Berlin (Germany). The emphasis of CAPDA is laid on the configurational development procedure, especially focusing on parametric studies and optimisation. The iterative synthesis uses input data from an extensive statistical data bank, and results in a computer representation of geometry and performance. An optimisation module based on a direct search strategy can be applied for the development of an optimised baseline configuration. In the first design phase emphasis is laid upon parametric variation of a large number of design variables rather than detailed analysis. Therefore the mathematical model is relatively simple, although all important design elements are included.

The initial baseline is obtained either from a statistical processor or as a ’guesstimate’ by the designer. Then the configuration development is performed by the design synthesis procedure, carrying out an iterative synthesis of several disciplines until all design variables have converged and all constraints are met. Later more detailed analysis procedures are carried out. The design synthesis is a pre-defined procedure in analogy to the classical design approach (standard design criteria). Some inequality constraints have been converted into adequate equations using appropriate secondary conditions, to reduce the numerical computational efforts to solve the problem. Simple iterative methods are applied. The design procedure is performed manually.

The geometrical model in CAPDA is described by parametric, analytical functions which describe the surface. This also reduces the data necessary to describe the model in the statistical database. The statistical database is used to choose proper initial values. It performs statistical evaluations such as variance, regression and correlation analysis and multiple linear approximation methods to calculate coefficients (’interactive statistics processor’).

Though the use of specified definition parameters simplifies the modelling of the aircraft, it also limits CAPDA to the design of conventional aircraft configurations. Some other issues which need attention according to [Haberland, 1984] are the possibility to change the sequence of the design process, the graphical presentation of diagrams and the integration of the statistics processor.

Within its successor VisualCAPDA, developed in cooperation with PACE Aerospace and Information Technology Company GmbH (Berlin), some of those shortcomings have been eliminated. VisualCapda features a graphical user interface instead of a Fortran code module, a new library concept which is fully based on dynamically linked and loaded modules, a larger operative flexibility through improved interactions, and an on-line information system which helps the designer to manipulate the design parameters and the selection of the analysis method from the methods library.

Other developments are in progress at PACE, focusing on a more general approach based on an object-oriented class library for aircraft components and characteristics. These developments started under the project name Flying Objects [Schneegans, 1998] and have resulted in a commercial package called Pacelab [PACE, 2000]. This object-oriented library forms the core of a product line which includes Pacelab Mission and Pacelab Cabin. With Pacelab Mission flight missions are calculated for the assessment of airframe/ engine performances. Pacelab Cabin applies knowledge-based techniques for the design of the cabin interior and emphasizes the flexible geometric modelling with pre-defined cabin items. The latest development is the Pacelab Design tool for aircraft conceptual design and analysis. Pacelab products are claimed to be used in the Airbus A320 and A330/340 programs.

CDCS: Configuration Design Computing System [Britton, 1987]

CDCS is a CAD system developed specifically for aircraft design at Boeing. Initially CDCS supports the design of wings and horizontal and vertical stabilisers, but extension to the design of bodies and engine nacelles were planned according to [Britton, 1987]. Three levels of design are supported: the initial sizing and shaping planforms, the placement of detail components on planforms, and the definition of the surface surrounding planforms. CDCS is parameter driven, so the user can enter higher level parameters such as the aspect ratio. Within the system geometrical relations have been defined, which result in an automatic updating of the model when one parameter is changed. The geometric definition of the lifting surfaces appears to be rather detailed in [Britton, 1987]. Most of the analysis calculations should be performed by external programs, which could communicate with CDCS through specialised interfaces. CDCS was used in the development of the Boeing 7J7 project.

The emphasis of the system was in 1987 on the geometrical modelling of the aircraft, starting with the individual components. The extensive use of parameters facilitates the model definition when designing conventional components. No recent publications are known.

PASS: Program for Aircraft Synthesis Studies [Kroo, 1988] [Kroo, 1992] [Kroo, 1996]

PASS is an "executive system and a collection of routines for analysis of subsonic transport aircraft" [Kroo, 1992]. It is developed at Stanford University and supported by NASA. The basic system consists of an executive routine which manages the communication between the database, the user and the analysis modules. The analysis modules contain short procedural routines and have fixed sets of input and output parameters. The user selects a parameter to be computed, and generates a path through the set of analysis modules which are necessary to evaluate a parameter. The routine searches for modules with the requested output parameter; when an input parameter of the module needs to be computed, the executive routine searches again for the appropriate module, etc.. This quasi-procedural architecture of the system is the key to its usefulness for complex design tasks. The difficulty is the complex interconnections among these modules that are difficult to manage. Iterations are recognised and automatically performed using initial values. The convergence of the numerical process can not be guaranteed, however. One- or two-dimensional parameter studies are possible, as is the integration with several numerical optimisation methods.

Another useful feature is the expert system which warns the designer when problems arise, for example when constraints are violated ("Take-off field length too large") or just simple design rules are triggered ("Fuselage-mounted engines usually require a T-tail") [Kroo, 1988]. The expert system also provides suggestions to the designer as to how the identified problems may be resolved or how the design might be improved. It uses a forward and backward chaining inference engine to examine the current database, the rule base and the warning strings posted by the warning system. The suggestions also consists of strings.

In [Kroo, 1992] more emphasis is laid upon the multiple-window graphical interface and the optimisation techniques. The user interacts with the system by viewing or modifying the database, conducting trade studies or optimisation, or generating graphical elements that describe the design.

Though not explicitly mentioned in [Kroo, 1988] and [Kroo, 1992], the geometrical modelling seems to have the same character as the simple analysis routines: a pre-defined, small set of input or definition parameters of standard aircraft components.

ADAS: Aircraft Design and Analysis System [Bil, 1988] [Bil, 1989] & Chaince [Blijenbergh, 1997]

ADAS has been developed at the end of the eighties at Delft University of Technology (The Netherlands). It is a generic preliminary design system: the analysis methods are considered as input to the system, similar to data. So the modules embody the domain knowledge and reside outside the system where they can be easily modified and reorganised by the designer and be tailored to a specific design problem. Initially the commercial Medusa[1] drafting and solid modelling system is used for the geometrical definition. Later this system has been replaced by

1. Medusa was a software product developed by Cambridge Interactive Systems (CIS) in the United Kingdom.

AutoCAD. These systems are also used to perform other graphical tasks such as plotting of diagrams.

In ADAS the designer begins with the definition of the geometry and some non-geometrical parameters. The designer can select a configuration from the database or define his own, preserving the strict organisation of the model which is necessary for the interface with ADAS. The set of input and output data is accurately specified for each module. In a batch-file the use of different modules can be assigned. The shell of ADAS offers a parametric survey mode and an optimisation mode. A large number of plots are possible to present the results.

The ADAS system is very capable of applying specialised modules. The central database requires an accurate declaration of the input and output parameters, however, which hinder to make maximum use of the flexibility. Due to this set-up of the modules, ADAS is a strongly procedural system.

Currently the ADAS system is not operative any more. The development of its successor Chaince [Blijenbergh, 1997], with a graphical user interface and object-oriented data structure increases the flexible use of the modules, is in a very initial stage of development.

A design program developed at TsAGI (former USSR) [Denisov, 1990]

Since the mid-seventies a dialogue system has been developed for preliminary design of passenger aircraft at the Central Aerohydrodynamic Institute TsAGI in the former USSR. The system is constructed according to the modular principle: each module contains domain knowledge about a specific area, and requires a fixed set of input data. Some modules require a component's geometry. The system contains a database with pre-defined components from which the designer can choose, for example between a one-deck or a two-deck body, with a circular or a double-bubble cross-section, etc.. The user must define a pre-defined set of input parameters, and select variable parameters which are subject to optimisation studies. The system can optimise the selected configuration using pre-defined objective functions.

[Denisov, 1990] emphasizes the optimisation results of the system. The structure of the system insinuates a more or less fixed computing path due to the specific input and output of each module.

Aircraft Configuration Design System at Cranfield Institute of Technology [Pasaribu, 1991]

In [Pasaribu, 1991] the development is described of a design system for civil transport aircraft, which includes the design of the aircraft configuration. The system is built of several modules or ’sub-systems’, each taking care of a specific design item such as aircraft configuration analysis, engine sizing, weight estimation, aerodynamic analysis, etc.. A separate ’system control’ module determines in which order the sub-systems will be called upon, depending on the input data and the data produced by the sub-systems. It uses logic reasoning to make these decisions. This data is stored and maintained in a central database. The database also contains a compilation of data of existing aircraft. Another special module is the graphical module that presents simple 2-dimensional drawings, such as cross-sections, to give visual feedback.

Most of the sub-systems use conventional computer techniques, except for one: the ’aircraft configuration analysis’ module. Similar to the system control module, the configuration analysis module applies logic reasoning. To select an appropriate aircraft lay-out, general rule-of-thumb knowledge is implemented in if-then-rules. When more than one configuration option is left, the user has to make the final choice. Attention is paid to the explanation of the reasoning, by presenting the chain of applied rules, and explanation is provided by text files with general characteristics that are attached to each configuration option. The case study in [Pasaribu, 1991] shows that the database with data of existing aircraft is used to provide initial parameter values. It seems that the user has to search for the relevant values himself; no attention is paid to automate the support for this activity. The system is implemented in the C++-language.

ACDS: Aircraft and spacecraft Configuration Design System [Yuan, 1992]

ACDS is a parametric preliminary design system for aircraft configuration that has been developed at the Northwestern Polytechnical University in Xi’an (China). By using only a limited number of model definition parameters, the geometric model is easily changed when a parameter is modified. The geometry definition begins with the separate components. Emphasis is laid upon the mathematical model of the component’s surfaces since this is important for the application of the so-called area rule, which is necessary for supersonic calculations. The system accurately calculates the intersections between the components.

In [Yuan, 1992] the geometry definition and modification techniques are highlighted. The designer has to estimate the initial values of the component definition parameters. Little attention is paid to the analytical methods; only the area rule is mentioned. ACDS is capable of visualizing the model in three-view drawings and wire-frame models, and visualizing the development of the design parameters.

Design Sheet [Buckley, 1992] [Gonda, 1992]

Design Sheet is developed at Rockwell International. It conveys the idea of a typical spreadsheet: parameter studies are easily performed by sets of algebraic relations. Unlike usual spreadsheets, however, the designer does not have to define which are input and which are output parameters when specifying the constraint network in Design Sheet. The computer automatically determines the computational plan for how to compute the model parameters from any valid set of input parameters [Buckley, 1992], which suggests a forward chaining inference engine (see the explanation in next chapter). This allows the designer to consider a wider range of alternatives during a design, because he is free to choose which input parameters to consider. Design Sheet provides extensive plotting support for trade studies, model browsing, both equality and inequality constraints, a limited amount of constrained optimisation, and first order error analysis. The algorithms are based on graph-theoretic ideas which are used to determine the order of algebraic and numerical operations and to decompose the constraints set into tractable subsets. When possible symbolic methods are used to solve the subsets, otherwise numerical methods are applied. The individual solutions of these subsets are then combined into a complete solution. Iteration cycles are called ’strongly connected components’ and are solved numerically. During creation of network the user is forced by necessity to make estimates of various parameters based on incomplete information. Design Sheet uses the Common LISP computer language.

Although [Gonda, 1992] shows some geometrical models of the design, this aspect is not worked out.

ACSYNT: AirCraft SYNThesis [Mason, 1993]

ACSYNT is an aircraft configuration sizing and optimization code, but can also be used for mission analysis. It was initiated by NASA some twenty-five years ago, but the continuous development efforts have been formalised in the formation of the ACSYNT Institute. In the late eighties a graphics interface was developed, implemented in PHIGS. With ACSYNT an aircraft geometry can be created using design parameters such as wing area and taper ratio instead of specifying 3-D points and curves. The data needed by the analysis modules is entered in a "spreadsheet" format. Each module deals with a specific discipline, such as costs. The spreadsheet has on-line help for all the analysis variables, formula capabilities, and can be custom tailored for individual customer’s needs. The analysis is implemented in FORTRAN and C computer languages.

ACSYNT provide ’starter’ or template files of geometry and input data for various aircraft types. These starter values are normally employed in initial analysis. In addition to internal estimates, ACSYNT also provides (geometry) interfaces to communicate with external calculations. The development efforts have been concentrated on the accuracy and liability of the implemented knowledge. Scaling factors are widely used. In [Mason, 1993] it appears that the design path is determined by the designer, based on the input and output parameters of the geometry templates and the analysis routines.

AAA: Advanced Aircraft Analysis [Roskam, 1989]
& G.A.-CAD: General Aviation Computer Aided Design [Anemaat, 1997]

AAA is an aircraft design and analysis code [Roskam, 1989]. The AAA program started out as a computerized version of Roskam’s *Airplane Design* books [Roskam, 1987-1990], featuring a user-friendly interface. The code has now been developed beyond that capability by the DARcorporation. Because of Roskam’s background, it has considerable capability in the stability and control area.

The code consists of several modules which cover specific disciplines such as the weight, aerodynamics, performance, geometry, propulsion, stability and control, flight dynamics, loads, structures and the costs. Each module can be used separately, requiring a particular set of input data and producing a pre-defined data set. When using all modules an aircraft is designed with more details than what in this thesis is considered an aircraft concept. The designer determines which module to use and when. The order of the modules is strongly directed by the sets of input and output data. Hence the designer has to have a good overview on the modules.

Every module has its own graphical user interface, represented by one or more windows. The designed geometry is also presented graphically, but only per component: the result of one module. There is not a central interface which shows the overall geometry of the aircraft. AAA can also plot the results of sensitivity studies.

Based on AAA is the G.A.-CAD system [Anemaat, 1997]. This system is also developed by DARcorporation. It differs from AAA that in G.A.-CAD the common database is split into two parts: a flight condition independent database and a flight dependent database. This procedure allows analysis of several flight conditions within the same project. Also new is an interface to a CAD program, which contains a surface modeller and has all standard PC-CAD capabilities. It is able to perform aerodynamic specific design tasks, such as wing planform, airfoil cross-section, wetted areas, centre of gravity, etc.. New developments are focused on the evaluation of innovative configuration design approaches; see also [Olsen, 1997].

RDS [Raymer, 1996]

RDS is an aircraft design, sizing and performance code. The program is the implementation of Raymer’s book [Raymer, 1992] and is nowadays marketed by Conceptual Design Corporation. It incorporates a CAD module to develop the geometry, as well as analysis modules including cost analysis and airline economics. Adjustment factors are available to change the code predictions to account for circumstances where the internal weight, drag and thrust estimates are not accurate. It has a relatively complete mission capability. Up to now there are four versions available for different purposes: the Professional version is much more comprehensive than the Student version; the Homebuilt version is tailored to the needs of homebuilt propeller-driven designs; and the ’EZ’ version is a stripped version to be used by anyone to estimate the performances from very few inputs.

In [Raymer, 1996] emphasis is laid upon the optimisation techniques. An optimisation module was developed using a simple yet robust scheme based on computer power rather than on more sophisticated strategies. Simultaneous optimisation of the key parameters is possible. It uses existing RDS-Professional analysis modules, coupled with a parametric optimiser.

2.5 Conclusions for the proposed computer support system AIDA

The computer tools in Section 2.4 support the first phases in aircraft design in different ways. Some are focused on the geometry generation of the aircraft and geometry related analysis, such as CDS, CDCS and ACDS. Generally they make use of the standard components as mentioned in Section 2.3.2. Defining the components with a small set of high-level parameters, such as the aspect ratio of a lifting surface, facilitates their design. On the other hand, this restricts their use to conventional geometries.

In most design support tools the domain knowledge is organised into separate modules or routines which communicate via shell structures. Much effort has been put in the reliability and robustness of the modules. Although this split up enhances the designer's overview of the domain knowledge, the many interactions between the routines reduces the transparency and makes a proper and flexible use difficult.

Other development teams have recognised these draw-backs and focused on the management of the routines. The systems Paper Plane, PASS and Design Sheet, for example, support the designer in determining the sequence of the routines. They automatically arrange the computing path on-line. Paper Plane and Design Sheet offer even more flexibility since they are able to invert the design functions, so that the functions can be used to calculate output as well as input parameters. These design functions embody only simple analysis routines.

Another area of support is the optimisation of the initial configuration, which is covered by most of the described tools. Different techniques have been implemented with different capabilities. Their usability depends on the robustness and the speed.

Only CAPDA supports the generation of the initial configuration, i.e. the *suggesting* task within the design process. In [Haberland, 1984] statistical methods are mentioned which have been implemented to find proper initial parameter values. The expert system within PASS is also used for suggestions, but only to *suggest modifications*.

The examination of these current computer design tools lead to wishes about the new support tool AIDA, the subject of this study. The aim is to integrate some of the good characteristics which are shown by the different computer tools, improve them where possible, and avoid the negative aspects.

One task the AIDA system should be able to do is to support the *suggesting* task. This means helping the designer to propose a proper initial aircraft configuration to form a good starting point for the next design steps. The current support tools show that this is one of the first activities within the design process, and only one tool (CAPDA) provides some assistance in this area.

Another item is the flexible use of the calculation routines or modules. Most of the current design tools have a modular set-up with each module containing calculation methods for specific disciplines. Although the contents are well developed, their interaction is restricted due to their specific sets of input and output parameters. This considerably ties the designer's hands on the use of the modules. Within the AIDA system emphasis is laid upon the flexible use of the modules, which should facilitate the first *prediction* and *evaluation* of the aircraft concept for a variety of configurations. The Paper Plane, PASS and Design Sheet programs show such a flexibility.

The geometrical components of the support tools have also proven their usefulness. Their visual feedback is very practical for understanding the designed concept; i.e. the *evaluation* task. Therefore the AIDA system should also contain such a component. To relieve the designer the aircraft model should be generated (semi-) automatically.

With the emphasis on the *suggesting* task and a flexible (first) *prediction* and *evaluation*, the AIDA project concentrates on the first design step: the generation of a feasible aircraft concept. For further elaboration of the concept the designer is expected to use the well-proven existing tools. Therefore relatively simple design relations are considered, such as those described in [Roskam, 1987-1990] and [Torenbeek, 1982]. The generated aircraft model should be suitable as input for those tools.

3 AI and design

3.1 Introduction

The main theme within this thesis is the application of AI techniques in design, i.e. how AI can play a supporting role in the design process. In this context it is interesting to know how knowledge can be applied to reason about design, i.e. how knowledge can support the *generation* and *evaluation* of design solutions. Therefore the characteristics of design knowledge and how it can be applied are surveyed.

A convenient view is to consider knowledge (not only design knowledge) as being composed of basic knowledge components. It is useful to illustrate these components by if-then-rules: *if* a condition is found to be true, *then* a conclusion can be drawn [Coyne, 1990]. Three elements are recognized when using such form of knowledge representation: the if-then-rule itself, representing the knowledge component, a situation A which satisfies the antecedent and a situation B which describes the consequent; see Figure 3.1. The knowledge component can represent mathematical knowledge as well as quantitative domain knowledge, but also meta knowledge, i.e. strategic knowledge about how to use the domain knowledge.

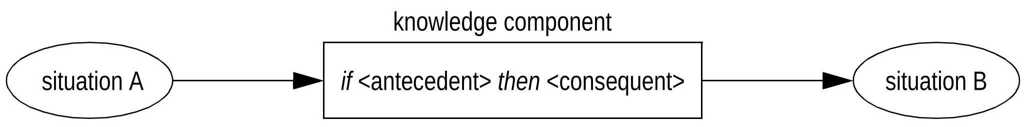


Figure 3.1. The three elements involved in the application of a knowledge component.

The arrows in Figure 3.1 indicate the direction for which the knowledge component is valid: using this rule situation B can be deduced from situation A. This is called deduction. However, when situation B is given, the rule can also be used to conclude that situation A has *probably* caused situation B. This *plausible* reasoning is called abduction.

The overall design process, proposing a new design given the design specifications, can also be seen as an 'abductive' process [Roozenburg, 1993]. The solution (the artefact description) can not be derived directly from the design specifications using domain knowledge only. However, when the description of the artefact is known, domain knowledge can be used to deduce the performances. Such knowledge is usually based on physics and accurate enough for verifying that the artefact description meets the desired functional specifications. This is illustrated in Figure 3.2.

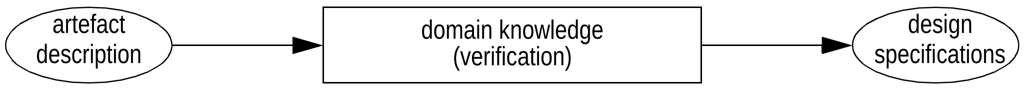


Figure 3.2. The use of domain knowledge in the design process.

We can now describe the role of several AI techniques.

The Rule-Based Reasoning technique (RBR) shows much resemblance with the scheme of Figure 3.1. Applied to design, it could be used to verify that a proposed solution satisfies the design requirements (Figure 3.2). The if-then-rules represent the basic domain knowledge components and a separate inference engine determines the strategy for using the rules. However, to transform domain knowledge into these if-then-rules can be difficult. It is a serious problem to collect a concise but comprehensive selection of domain knowledge from experts and put this into causal relations. Common sense for example is typically hard to formulate in rules, while mathematical reasoning is more easily formally expressed.

Therefore the RBR technique seems useful for design tasks where mathematical relations are involved, such as the prediction and evaluation tasks of Figure 1.1. In Figure 3.3 the use of such rules in design is schematised. Because rules usually represent simple knowledge components, many rules are required. One issue of RBR is to choose the appropriate rules which will lead to the desired relation between the artefact description and design specifications. This issue is less difficult when the domain is well-defined and well-structured.

The RBR technique is described in more detail in Section 3.3.

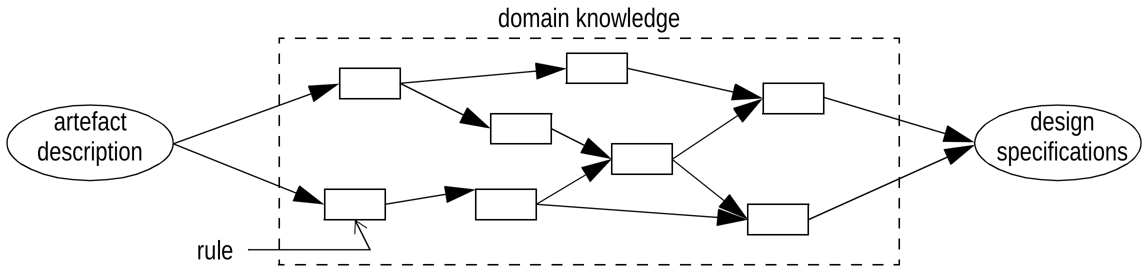


Figure 3.3. Implementation of RBR techniques in design: deducing specifications from artefact description with mathematical relations.

Other AI techniques, such as Artificial Neural Networks (ANNs), capture the knowledge in a different way. ANNs focus on the automatic capturing of the knowledge component, which is called induction [Roozenburg, 1993]. ANNs are used when explicit domain knowledge is not available, e.g. when relations between data are too complex or not explicitly known. An ANN acts as a black box representing the knowledge component of Figure 3.1.

An ANN consists of a network of nodes connected via adjustable weight factors. By tuning these factors the ANN is trained to exhibit a desired behaviour, e.g. produce a pre-defined output from a specified input. The ANN is trained with a large set of examples which provide the proper input-output combinations. The training is concluded when the desired output can be generated for all examples with a certain accuracy. A well-trained ANN can be used as a knowledge element. As a consequence of the methodology this element represents generalized knowledge or tendencies between problem and solution without making the captured knowledge explicit. The ANN can be used to deduce a new structure for a new function description when it has been trained for that purpose. This is symbolized in Figure 3.4.

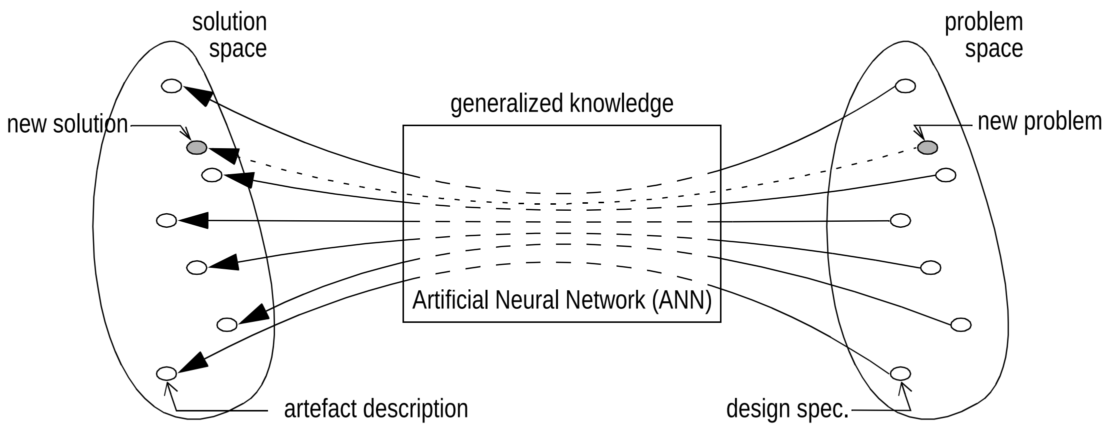


Figure 3.4. The representation of an Artificial Neural Network (ANN) as a generalized knowledge component between design problem and solution.

This generalized knowledge has a global character, meaning that individual differences and extremities are smoothed out. An ANN is not able to solve ’new’ problems which differ much from the problem-solution combinations in the training set. Hence a certain degree of homogeneousness of the problem and solution space is assumed, without discontinuities. These premises make ANN less suitable for design, where these spaces are seldom continuous. Another disadvantage is that, given the complexity of the design problem, the training sets will need to be extremely large. Hence, ANNs are not studied further in this chapter.

Another AI technique is Case-Based Reasoning (CBR). While an ANN generalizes and classifies the domain knowledge, the CBR technique (re-)uses local domain knowledge: CBR links the problem and the solution at the level of single cases. CBR implements reasoning by analogy, meaning that new solutions are proposed based on similar problem descriptions of existing cases. This method assumes only local continuity of the solution and problem spaces in the neighbourhood of the cases; globally these spaces may contain discontinuities. The complex knowledge component within a case, which relates the solution to the problem e.g. the structure to its function, is reused without making it explicit. Figure 3.5 illustrates this.

CBR techniques provide a link between problem and solution, or function and structure, so it can be used to deduce an artefact description given the design specifications. Unlike ANNs, though, the CBR techniques use *local* domain knowledge and don't need so much cases as ANN requires for proper training; considering the complexity of the design cases. These are valuable assets which suggest that CBR might be a useful technique for design to propose a configuration from design specifications. Especially since explicit knowledge is sparse or too qualitative at this phase and generalized (domain) knowledge is not specific enough. The technique is further described in Section 3.2.

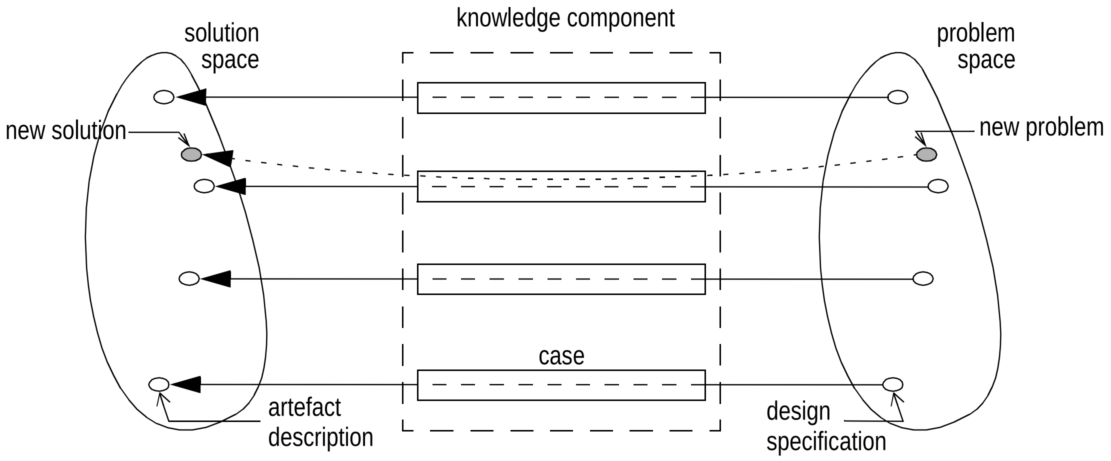


Figure 3.5. The representation of the Case-Based Reasoning (CBR) techniques with local domain knowledge components.

Finally, another technique can be mentioned which is often associated with AI: Genetic Algorithms (GAs). Unlike the previous mentioned AI techniques, GA is an optimisation technique. It forms new combinations out of existing artefact descriptions of cases with good specifications and anticipates that some of the many newly created artefacts may perform better than the existing ones. Though the determination of the overall quality of a case function requires a good understanding of the domain knowledge, the combining process may work more or less randomly. Therefore many combining sessions are necessary to make progress. As an optimisation technique, the basics of a GA are hard to illustrate with the elements of Figure 3.1.

One of the major issues of GAs is the formulation of the merit function to determine the quality of a solution. This requires a certain level of quantitative knowledge about the domain, which is hardly available at the conceptual design phase. Therefore this technique has not been considered within the AIDA project.

Another issue that is described in this chapter is the geometrical representation in Section 3.4. This is not an AI issue but a significant design issue. The applied geometrical representation technique is critical for the flexibility of the aircraft concept model, for example parameterised modelling.

3.2 Case-Based Reasoning

3.2.1 Introduction

The Case-Based Reasoning (CBR) methodology has its origins in the late 1970s, begin 1980s. A new view on human reasoning was proposed which is based on analogical reasoning [Schank, 1977]. Schank introduced the dynamic memory theory which assumes that remembering, understanding, experiencing and learning are not separate issues but highly interconnected. Schank suggested that this dynamic memory is organised into Memory Organisation Packages (MOP) which contain certain episodes or experiences. MOPs are interconnected with each other by explanations of the situations and expectations about the normal progress of events. When a new problem has to be solved, a relevant MOP is retrieved to come up with a solution based on previous experiences, instead of deducing a solution completely from scratch with a set of general relations. When a new solution has been found, the problem solving event can be stored as a MOP and added to the memory.

This way of reasoning eludes some of the difficulties with RBR, such as the elicitation of domain knowledge. With reasoning by analogy the domain knowledge is implicitly embedded in the cases and does not need to be fully understood. Because CBR deals with partial similarities between a new problem and previous problems, it is also possible to handle incomplete cases or problem descriptions. Another advantage of this type of reasoning is that it allows qualitative reasoning about similarity of the problem, while still a quantitative solution can be produced. This seems to be a more natural way of reasoning and may considerably speed-up the process of problem solving; in addition previous failures may be avoided.

Basically, the working of CBR can be illustrated with Figure 3.6. Before the CBR methodology can be applied, a case-base has to be created. The case-base contains previous solved problems in the form of cases. In each case a description of the problem and a description of the solution is encoded. For a proper working of CBR enough cases should be collected to cover the problem and solution spaces well.

The first step in the CBR methodology is to translate a new problem into the vocabulary of the problem description. Next a case is searched for with the most similar problem description. The solution contained by that case is assumed to be an appropriate starting point for the new solution. The suggested solution is then modified in order to better satisfy the original problem description. After evaluation and verification the new "problem-solution" combination can be considered as a new case again.

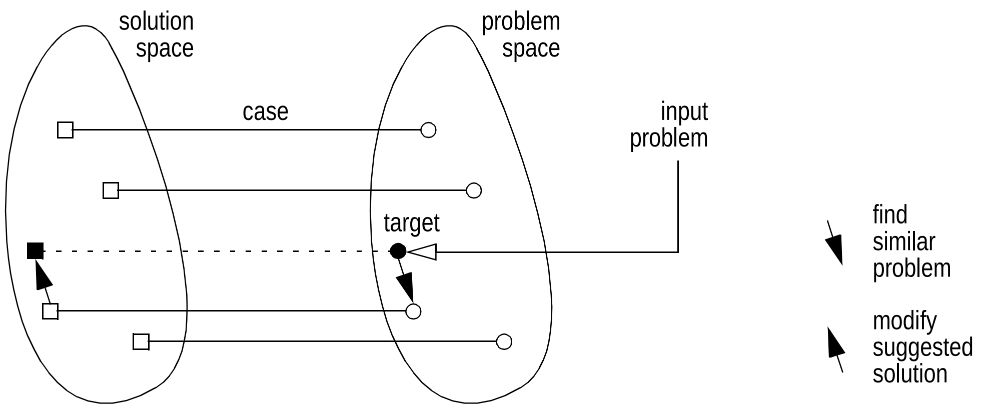


Figure 3.6. Principles of problem-solving with Case-Based Reasoning (from [Leake, 1996]).

The productive reuse of previously solved problems is based on three assumptions [Kolodner, 1996] [Jarmulak, 1999]:

- regularity of the world: when identical actions are executed under identical conditions, the outcomes will also be similar;
- local consistency: locally, a small change in the problem description will require only small changes in the solution; globally discontinuities may exist;
- ease of adaptation: small differences between problems will be easy to compensate for.

The CBR methodology is used in many areas, such as help-desks, classification, diagnosis, teaching, planning and design. In this thesis the CBR methodology is applied to design problems, in particular aircraft design. Typical for this application area is the relatively small number of previous cases and their complexity, i.e. they may contain many parameters which are related with each other in unclear ways. Therefore special attention has to be paid to the description of the cases; see Section 3.2.7. Related with the complexity is the search for relevant cases. Together with other important issues such as case indexing and case adaptation, these subjects are described in more detail in Section 3.2.3 to Section 3.2.6. First the CBR basic cycle is discussed in next section.

3.2.2 The CBR cycle

There are several central tasks that all CBR methods have to deal with. [Aamodt, 1994] suggests four RE's:

- REtrieve the most similar case or cases;
- REuse the information and knowledge in that case to solve the problem;
- REvise the proposed solution;
- REtain the parts of this experience likely to be useful for future problem solving.

Together these tasks form the CBR cycle as sketched in Figure 3.7. The cycle starts with the definition of the new problem, i.e. the target. This requires a clear definition of the vocabulary in which the cases are expressed. A separate section deals with the issues of what kind of information the cases should contain and how it can be organised (Section 3.2.7). Only when the case representation has been fully elaborated, the case-base can be filled with cases.

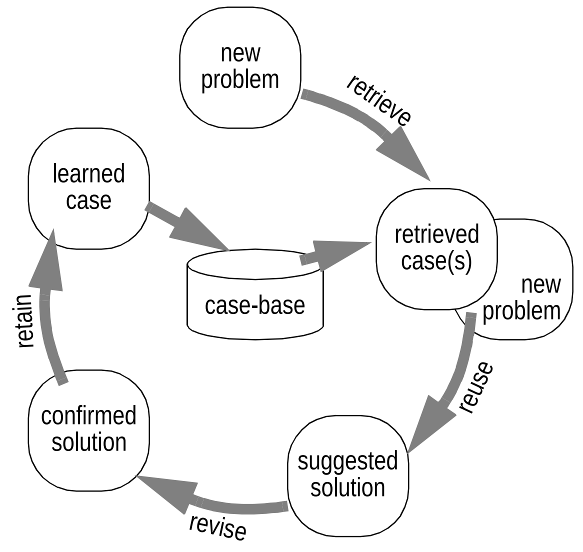


Figure 3.7. The CBR cycle (adapted from [Aamodt, 1994]).

Retrieve

With an appropriate problem description, one or more relevant cases are retrieved. The selection of these cases is based on the similarity with the defined target. The similarity is measured locally and globally. Local similarities are based on the matching of the individual features. Together these local similarities determine the matching score of the complete case: the global similarity. The type of measurement of the global similarity determines how two partial matching cases can be compared. Several schemes can be used to measure these similarities.

From the similarity ranking list one or more interesting cases are selected. Although the case with the highest ranking is likely to be selected first, the user may also prefer other cases, for instance because they are more amendable to optimisation. Hence the matching criteria do not have to be identical to the selection criteria.

To speed up the calculation of the similarity measurements and still guarantee a thorough search of the case-base, the memory structure of the case-base has to be organised well. The organisation is implemented in a so-called indexing network. This network should be discriminating enough to allow proper selection of cases with high potential, but not too differentiating to exclude some potentially suitable cases.

To be effective, the case-base should contain cases which cover the complete design space. It is obvious that if the case-base only covers parts of the design space, solutions will never be suggested which are not available in the case-base.

See further Section 3.2.3 to Section 3.2.5.

Reuse

While the selection of the cases is based on the similarity score with their problem part, their solution parts are reused to create a feasible solution. There are several reuse strategies: to copy the complete solution part of one of the best-matching cases, to combine the solution parts of several cases, to extract an average solution from the selected cases, etc.. The process of changing the solution part for the purpose of reusing is called adaptation. The goal of adaptation is to suggest a better solution before detailed and expensive evaluations are performed. By adapting a case, the complex relationship between its problem and solution part, which is illustrated by the knowledge component in Figure 3.5, is violated. Therefore it requires a good understanding of the domain knowledge to perform proper adaptations. This is one of the major issues in CBR and is described in Section 3.2.6.

Revise

Only when the suggested solution is fully elaborated and placed into the real world, it can be properly judged. Generally revisions are needed to modify the suggested solution for the real world situation. This results in a solution which can be confirmed by practical use or by accurate evaluation tools.

Retain

The combination "problem - confirmed solution" represents a new case. By adding this new case to the case-base, it can be reused for future problems. This is considered to be a way of learning. One should be aware of the possible risks of this incremental learning method, though. The case-base could get biased by cases with too much resemblances.

By adding the revised, new case to the case-base, the CBR cycle is closed. The case is available for retrieval and reuse to solve new problems, facing the same issues again. The most important issues are described in next sections.

3.2.3 Case indexing

The case indexing structure provides the means to select the most suitable cases for further matching. The efficiency of the indexing influences the speed of the matching process, while the accuracy aims at its completeness, i.e. that each similarity is found. Case indexing comprises the organisation of the memory structure and the labelling of the cases and their features. It also embraces the case representation, but that aspect is dealt with in Section 3.2.7.

Several memory structures are feasible [Jarmulak, 1999]. The most simple one is the flat structure, where the indexing is synonymous with the parameters describing the cases. It requires the complete case-base to be searched for matching features, which leads to long retrieval times when the case-base is large. Its simplicity makes it easy to add and delete cases, though.

To reduce the search time clustered memory structures have been proposed. Groups of similar cases are stored in clusters and represented by some prototypical cases. Such a prototypical case can be an abstract case or a real representative case. The matching process concentrates on the similarity with those prototypical cases, and therefore is much faster. A disadvantage is that future similarity interests have to be predicted to organise the appropriate clusters. It is also more difficult to add cases.

Another memory structure is the hierarchical organisation. This structure can be based upon the grouping of cases which share the same features, called shared networks, or upon discriminating features which guide for example decision trees: discriminating networks. These networks can be combined with weight factors to bias the search process. The hierarchical organisation can become very complex which increases the costs for adding cases.

In general the retrieval process can be speed up by parallel search methods.

3.2.4 Case matching

The purpose of the matching procedure is to rank cases according to their relevance to the problem. The ranking is based on the similarity measurements between the cases and the problem description or target. An important issue is to decide on which parameters the similarity will be measured. Another is how to measure the similarities. The measurements are built on two components:

- the dimensional match: the degree of matching along each dimension;
- the aggregate match: the weighted sum of each dimensional match of a case.

There are many possibilities to define the matching criteria. For the definition of the dimensional match, for example, several distance measures between feature vectors are available. Things can get complicated when Booleans, discrete scalars, texts or images are involved. By aggregating the dimensional matches the cases can be compared with respect to the target. This can be a difficult aspect, even without trying to formalise the aggregation method. The determination of weight factors is rather arbitrary and may result in unexpected matching rankings. Also the problem of incomplete cases should be answered, i.e. how to deal with missing features?

Often it is easier to judge the solutions than to compare their problem descriptions. Therefore it may be useful to put more effort in the case selection instead of refining the matching criteria for the problem description. In that situation several best matching cases are retrieved and the user has to select the most appropriate solution(s).

3.2.5 Case selection

As has been mentioned before, it is not always the best strategy to select the case with the highest similarity score to provide an initial solution. The goal of the selection is that from the selected case(s) a solution can be formed which can easily be adapted to completely satisfy the problem. This does not have to be the best-matching case. It depends on the reuse strategy which case(s) to select. Two approaches are possible:

- reuse one complete case solution
- reuse multiple case solutions

With the first approach the case can be selected automatically with the aid of the matching score, or manually from a pre-selection of best-matching cases. The other approach also requires a pre-selection of cases, which is usually determined by the similarity ranking. To combine these cases one can focus on the similarity between the cases, and reusing for example highly similar components, or one can focus on the differences between the cases, and integrate different components. So ranking the cases can be based on the similarity of complete cases or on local similarity scores, such as the matching of case components or individual dimensions.

How the conversion of the cases is performed in order to propose a feasible solution is described in the next section.

3.2.6 Case adaptation

Case adaptation concerns the combining of selected cases or performing other case adjustments. The purpose of adaptation is to create a better solution for the new problem than can be found in the available cases. Adaptation differs from more straightforward modifications (parameter tuning) because it is based on the use of (parts of) alternative cases with their specific problem-solution combinations; i.e. a form of reasoning by analogy is applied. The intention is to make rough improvements to the suggested solution. A good understanding of the problem domain is needed to evaluate the matching results and to suggest appropriate adaptations. Due to its complexity case adaptation is one of the major research issues within CBR.

Three categories of adaptation strategies can be distinguished [Netten, 1997] [Jarmulak, 1999]:

- substituting
  Complete sub-solutions can be substituted by sub-solutions of other cases, for example case components can be combined. This adaptation method is suitable for design problems where each function can be linked with one or two components. When one function of the best-matching case does not match the target, the component is substituted that is thought to be responsible for that function.
- transformation
  When the interaction between parameters or components is more significant, adaptation by substitution only is not sufficient. In addition adaptation rules can be applied. With these rules the solution parameters are found which cause the mismatching with the case problem. Then modifications of these solution parameters are suggested. Different techniques can be applied to perform these tasks, such as RBR and Constraint-Based Reasoning. The adaptation can also be performed manually by the user instead of automatically, since adaptation knowledge is still difficult to formalise.
- derivational replay
  Another possibility is to replay the reasoning that led to the solution, instead of reusing the case solutions directly. This requires that cases contain the steps that were made to solve the problem. A new solution is derived by following the steps of the retrieved case for the current problem. The idea is that similar problems require a similar set of decisions to be taken.

To perform the adaptation process some basic issues have to be addressed. It depends on the adaptation category which issues will need more attention. The fundamental questions are:

- Which parts of the retrieved case(s) do not meet the problem definition?
  In design problems this usually concerns the determination of the mismatching functions of the selected case.
- Which parts of the suggested case solution cause the differences?
  This question deals with relating the mismatching functions to their responsible structure components.
- How can these solution components be adapted?
  When the responsible solution parts have been determined, the next question is how they can be adapted so that the similarity of the mismatching function will increase, i.e. a better solution is created.

- What are the effects of these local adaptations on the overall solution?
  While the goal of the adaptations is to improve the original case, they may have side effects. For example the adaptations of some parts of the structure can influence the functions which already did match the target. What are these influences, good or bad?

The last question points out an important consequence of adaptation: that the integrity of the adapted case may be violated. Each case in the case-base relates a problem to its solution. When parts of the solution are adapted, these relations are broken. The relations could be restored by applying transformation rules (the second adaptation category; see above), but to be practical these rules are usually kept too simple for a thorough evaluation. Because in design the relations between problem and solution are in general very complex (the reason for using CBR in the first place), the case will not be consistent anymore after its adaptation. Hence the importance of the revise task.

This consistency issue is especially important when the topology of the design is changed by adaptation, since its impact may be large. The influence of parameter adaptation is typically less significant.

3.2.7 Case contents

The usefulness of a CBR system strongly depends on the contents of the cases. This implies *what* is recorded in the cases as well as *how* it is recorded. The more relevant information the cases hold, i.e. *what* is recorded, the more beneficial the CBR can be. However, this information can only be well exploited when the information is properly encoded and organised (*how* it is recorded). There is a strong interaction between those two aspects: the case contents and its structure.

The contents of cases can be described in different ways, depending on the point of view. Each view angle will emphasize another aspect of the case data. In this section several points of view are described which seem interesting for applying CBR techniques within the design process.

With regard to case retrieval

With regard to the retrieving task within CBR, [Kolodner, 1993] divides the case contents into three components:
- a problem description;
- a solution;
- an outcome.

The matching of the case is determined by the similarity of the problem description. This component contains the goals and functions to be satisfied by the associated problem. It may also include a description of the context of the problem which was relevant for achieving the goals.

For design purposes the problem description is represented by the requirements, c.q. the performances or functions of the artefact.

With the solution component the solution for the new problem is derived. Two categories of CBR applications can be distinguished which are based on two different types of solutions:

- transformal CBR: reuse the *solution* of a previous case;
- derivational CBR: reuse the *method* of a previous case that resulted in the solution.

The latter kind of CBR system is useful for planning problems. Also design problems may be solved with derivational CBR when the purpose is to reuse the design steps, including the tasks, operations and strategic decisions. The other CBR category, transformal CBR, has been applied more often. In those applications the solution component contains information about the solution itself, and sometimes also background information such as acceptable solutions that were not chosen or unacceptable solutions that were ruled out.

In this thesis transformal CBR is applied since the design steps that lead to the development of existing aircraft, which is the application domain, are not (publicly) known. For this domain the solution component contains only the information necessary to produce the artefact: its dimensions, sizes, materials, production requirements, etc..

The third component, the outcome, describes the impact of the solution to the real world problem, e.g. how successful the solution was. It represents feedback for the solution. It may also contain explanations of failures, information about repair strategies, and possibly a pointer to another case which contains a next attempt to solve the problem.

The outcome can be extended with evaluation procedures that allow testing of case solutions in new situations. When cases do not include the outcome part, explicit feedback is missing. However, some feedback can be added implicitly by omitting unsuccessful cases in the case-base.

With regard to the data classification

In Chapter 1 the data classification of Function-Behaviour-Structure has already been mentioned [Gero, 1992] [Maher, 1995]. This classification structure has proven to be very helpful in analysing the design process, with the function data representing the (functional) design specifications and the structure data representing the (structural) description of the object. Hence this classification will assumable serve the analysis of the case contents. Therefore some more words about the Function-Behaviour-Structure (FBS) triad are in place.

[Rosenman, 1994] illustrated this data classification by naming the function as *what* (it) *does*, the behaviour as *how* (it) *does it* and the structure as *what* (it) *is*. The function represents the intended or specified functional features of an artefact, the structure represents the properties of the artefact that describe its physical existence, while the behaviour links these two by representing the set of characteristics that allow the structure to achieve its functions. For example the structure of a coffee cup describes the material, its shape and the dimensions. The behaviour of the cup is that it is waterproof, stable, smooth and that it insulates. Its function is that it can hold a certain volume of hot liquid. [Gero, 1992] points out the importance of behaviour: to disconnect direct links between function and structure in order to make the selection of a structure less intuitive and more reasoned. A difficulty of using the behaviour class is that the distinction between function-behaviour and behaviour-structure is not always clear.

[Rosenman, 1994] presents an additional class, called ’purpose’, which describes the goal of the artefact: *what* (it) *is for.* The purpose introduces the human aspect to the system, representing the impact of the function and behaviour. For example the coffee cup is for drinking hot liquids, but it can also be used to measure liquid and solid volumes. Hence, purpose is a multiple-view function. Excluding the purpose from the case contents prevents it to hold subjective information.

In addition to the function-behaviour-structure triad a function-means tree can be applied. This structure supports the decomposition-integration approach of design. In this approach the specified functions are first decomposed into simpler functions, until basic functions are derived. These basic functions can directly be linked with basic structure components, which are integrated to generate the complete solution to the problem.

This function-means tree introduces the notion of detail levels of the data, which can also be applied to the Function-Behaviour-Structure classification method. For example the range and the wing area are a function and a structure parameter of the aircraft, respectively. A detail of the aircraft, such as a bolt, also has a function and a structure: e.g. its connectivity function and its dimensions, respectively.

So, in addition to the problem-solution classification, which can be formalised by the FBS triad, data can also be classified by their level of detail.

3.2.8 Case representation

The way cases are organised in the case-base is essential for a proper indexing of the case-base, in order to find relevant cases quickly and accurately. This data representation topic is mainly an implementation issue. However, ordering the case information helps to analyse it, i.e. increases the understanding of the domain. So defining a case representation can be seen as a knowledge acquisition instrument.

Hence, the information and its structure can be meaningful for the user as well as for the implementation of the CBR system.

A wide range of representation formalisms are available to implement the case information, such as frames, semantic networks, predicate notations, rules and simple attribute-value combinations. Which formalism to apply depends on the type of information the cases should contain and on its intended use. When much interaction with the user is expected, the information must be understandable for humans. A graphical representation of the solution, for example, can be very clarifying. Another issue is the use of decomposition within a case, when cases are complicated. The decomposition may be helpful when adaptation strategies are implemented which are based on substituting parts of the solution.

For the geometrical information a similar kind of procedure can be followed. A case may contain the geometrical model of the design itself, the modelling plan which contains all modelling procedures, or the key parameters from which the model can be derived according to a standard procedure. The latter approach allows higher level reasoning and requires less memory space in the case-base, therefore may be easier for retrieval. It requires well-defined model templates outside the case-base, however. These aspects are described in Section 3.4.

In the AIDA system the information formalism is chosen with the focus on the effectiveness of the CBR module, not on the transparency to the user. The aircraft information is defined by a relatively small set of parameters, to keep the complexity of the case data low. However, these parameters contain all information which is required for the case retrieval task within CBR *and* for the definition of the complete aircraft concept. The Geometrical module takes care of the deduction of the three-dimensional model from the parameter set.

Only numerical and coded information is stored in the cases. The coded parameters represent the discrete information of the aircraft concept, such as the type of engine and its configuration. Via an object-oriented structure more information about a specific component type can be stored.

3.2.9 Reflections upon CBR and design

Analogy between the CBR cycle and the design cycle

The CBR cycle resembles the design cycle of Figure 1.1. This is not accidentally, since reasoning by analogy is a natural way of reusing knowledge; it is common to use experience gained with the solving of previous design problems for the generation of new designs.

The CBR methodology is more than just one single AI technique. In fact, "the CBR paradigm covers a range of different methods for organizing, retrieving, utilizing and indexing the knowledge retained in past cases" [Aamodt, 1994]. When CBR is referred to as one single AI technique, the tasks of retrieving and reusing are implied as well as the retaining task. Many other techniques such as Rule-Based Reasoning (RBR) and Genetic Algorithms (GAs) can be applied to aid these tasks; for example reasoning about the requirements, performing flexible similarity measurements, applying rule-base adaptation and evaluation, etc.. [Maher, 1995] refers to such systems as hybrid CBR systems.

CBR and creative design

In literature there is much discussion about the ability of CBR to support creative design. In general, three types of design can be distinguished [Dym, 1994]:

- routine design
  With this type of design all necessary knowledge is known in advance. It is known how to decompose the problem definition and what design plan to follow to solve the sub-problems. The difficulty is to choose the best components from a usually large solution space, and how to cope with the interactions amongst them. [Coyne, 1990] refers to this type of design as prototype refinement, as it deals with the instantiating of variables of a prototype. [Brown, 1989] calls this a class 3 design.
- innovative design (a form of non-routine design)
  This is a type of design between routine and creative design. The design is based on previous designs, but the modifications are so significant that new problems are introduced which require original solutions. The domain knowledge is known but the problem-solving strategy not. Known components or sub-solutions are integrated into an existing variable structure [Clibbon, 1994]. Other labels for this design type: prototype adaptation [Coyne, 1990] and class 2 design [Brown, 1989].
- creative design (another form of non-routine design)
  With creative design new components and new variables are introduced. The common domain knowledge does not cover the new (sub-)problems and appropriate problem-solving strategic knowledge is missing. [Coyne, 1990] calls it prototype generation. In [Brown, 1989] it is labelled a class 1 design, and is scarce since it requires major inventions and will lead to completely new products.

It is clear that CBR is a way to support at least routine design. When the adaptation task is properly implemented, it should also be possible to support innovative design. Components of previous designs can be combined to generate new solutions, possibly creating innovative designs.

The ability of CBR to support creative design is disputed. When a creative design is generated by using ’first principles’, i.e. deep knowledge about the domain, then there is no role for CBR. When other approaches have been followed, such as combinatorial design, analogical design, design through mutation [Rosenman, 1993], some support should be possible considering the similarity with CBR. Others argue [Logan, 1993], however, that the three design approaches mentioned by Rosenman can not lead to creative designs, hence CBR is not usable for creative design.

Concluding, routine and innovative design have the potential to be supported by CBR, whereas for creative design this seems to be impossible; discussions about this can be brought back to discussions about definitions. Another design category which is introduced by [Coyne, 1990] is 'configuration design', which lies between routine and innovative design. Because the conceptual design of (conventional) aircraft could be classified as configuration design (see Section 1.2), CBR seems to be a useful supporting technique for the AIDA system.

3.3 Rule-Based Reasoning

3.3.1 Introduction

In the Introduction, Section 3.1, the principles of Rule-Based Reasoning have already been discussed. By applying knowledge elements in the form of *if*-antecedent-*then*-consequent, new facts can be deduced from existing facts; see Figure 3.1. Each *if*-*then*-rule represents a small piece of knowledge in an explicit way. This distinctively differs from Case-Based Reasoning, where cases are used to represent large chunks of knowledge in an implicit way.

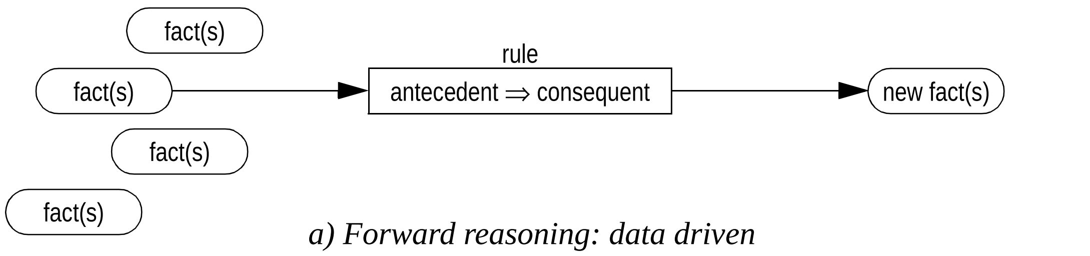
The knowledge expressed by the rules is largely domain dependent. Rules contain knowledge about a particular domain which is elicited from experts; hence the name Expert System which is strongly associated with such systems. Strictly spoken, a Rule-based system is a specific type of Expert System. But because other types of Expert Systems can be considered as variants of the Rule-based system, both terms are used promiscuously in this thesis. Those variations mainly involve the format of the knowledge elements, i.e variants of the rule as the knowledge carrier; see Section 3.3.3.

One of the ideas behind the development of Expert Systems was to create a tool that would contain knowledge from several experts, therefore being less dependent on individual experts, preventing bias and performing more consistently. However, it turned out to be extremely difficult to elicit knowledge from experts. For instance to extract all aspects of the antecedent of a rule, such as its context, appears to be very hard. As a consequence, an Expert System is usually valid for very restricted applications only. Knowledge acquisition is therefore an important issue within RBR.

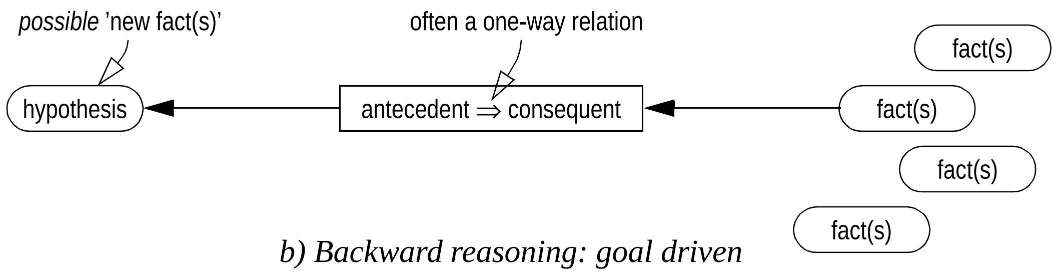
Another issue is *how* the rules (or alternative knowledge carriers) should be selected and combined. Complex relationships can be created by building a network of rules, i.e. combining the elementary knowledge elements such as sketched in Figure 3.3. A major difference with conventional programming techniques is that with RBR the sequence of applying the rules is flexible. This makes the testing of an Expert System rather complicated, since unexpected combinations of rules can give surprising, undesired results. Verifying the validity of the individual rules only is not enough.

Basically, linking the rules is based on matching the antecedent-part and the consequent-part of different rules. Subject to the similarity between the facts and the antecedent-part, a rule is selected. The consequent-part of the rule generates new facts; see Figure 3.8a. When the new facts coincide with the antecedent of another rule, this rule can also be selected and coupled with the other selected rule. The execution of the new rule may lead to other new facts, which can trigger other rules, etc.. This way of using rules is called forward reasoning or deduction, and is data-driven or ’bottom-up’.

The other way of using rules is called backward reasoning. Starting with a goal, a rule is searched for whose consequent-part matches the goal. When such a rule is found, its antecedent-part is suggested to be true. So, *possible* facts or hypotheses are abduced from existing facts with backward reasoning; see Figure 3.8b. These possible facts can lead to new hypotheses when more rules are linked in the same way. This is a goal-driven or ’top-down’ process, contrary to the data-driven process of forward reasoning.



a) Forward reasoning: data driven



b) Backward reasoning: goal driven

Figure 3.8. Forward and backward reasoning with if-*antecedent*-then-*consequent rules.*

The uncertainty of the newly proposed facts is typical for backward reasoning, unless the rules are valid in both directions such as arithmetic expressions. Hypotheses can be tested by applying forward reasoning with other rules.

In the process of linking the rules, three basic operations can be distinguished, which form the *recognize-select-act* cycle [Boullart et al., 1992]; see Figure 3.9. By matching the facts with the rule's antecedent, suitable rules are *recognized* and stored in an agenda. From the agenda the most appropriate rule is *selected*. The cycle is closed by the generation of new facts as a result of the execution of the selected rule: the *acting* task. These new facts may lead to new rules to be put in the agenda or old rules to be erased. Together the selected rules make the network. The generation of the network is finished when all rules of the agenda have been fired, i.e. activated.

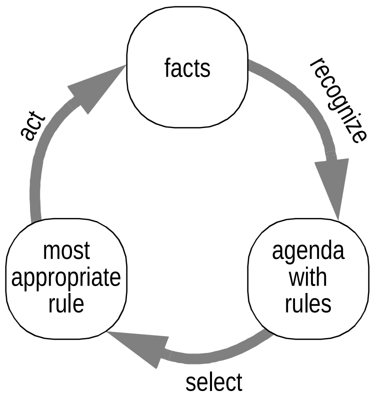


Figure 3.9. The typical recognize-select-act cycle of deduction with a Rule-based reasoning system.

3.3.2 The RBR components

In backward as well as forward reasoning the control of chaining the rules is clearly separated from the domain knowledge itself. Usually, a RBR system consists of three distinct components [Gevarter, 1987]. Their relation is sketched in Figure 3.10:

- knowledge base, which contains domain knowledge of experts;
- inference engine, which controls the domain knowledge;
- user interface, which provides the communication with the user.

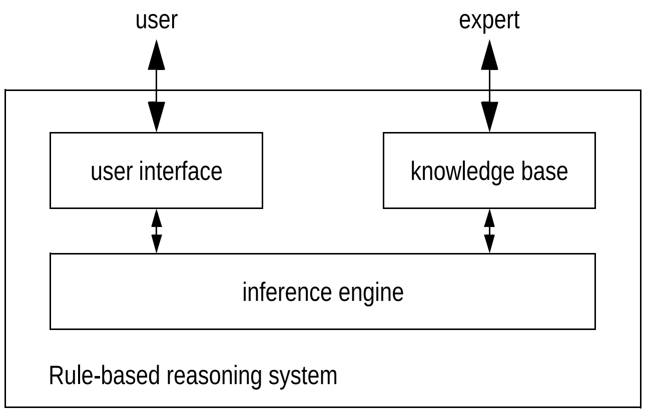


Figure 3.10. General set-up of a Rule-based reasoning system

The knowledge in the knowledge base is elicited from experts. It contains all the knowledge that is required for a specific application area. As has been mentioned before it often proves to be difficult to make that expert knowledge explicit and to formalise it. The basic format to represent the knowledge elements is the if-then-rule, but other forms are also possible; see Section 3.3.3. The knowledge can simply be documented by adding comments to each element.

The inference engine controls the use of the knowledge elements in the knowledge base. It is independent of the domain. The inference engine takes care of the *recognition* task of Figure 3.9. A major issue is to perform the recognition task effectively instead of checking every rule when new facts have been produced. Therefore several search strategies have been developed, each with its specific character with regard to robustness and completeness; see Section 3.3.4. That section also deliberates about the possibility to deal with uncertainties.

To perform well, the inference engine also maintains the actual memory status. It recollects which facts have been defined and which have been derived, which rules are on the agenda and which have been fired or skipped, etc.. This is important to re-run the reasoning process when the user has come to a dead end, for example because a less appropriate rule has been fired.

The *action* task in the RBR cycle of Figure 3.9 concerns the solving of the relations which are represented by the (set of) rules. Sometimes the set of relations is solved after the network of rules has been completed. However, the solving process may also be performed simultaneously with the generation of the network of rules, for example when the *recognition* task depends on the parameter values of new facts. Therefore a solver is often integrated with the inference engine.

Via the user interface, the user is able to control the inference engine, and to *select* the most appropriate rule of the agenda when this is not automated (Figure 3.9). The user interface also displays the activities within the RBR system, which is necessary for the user to understand the reasoning process and give the appropriate input. The user interface should eliminate the "black box" image of the system, which is important to be accepted by new users.

When the *selection* task is automated, the RBR system assumes a "closed world" situation. That is, the knowledge base is supposed to cover all the knowledge required to solve a problem. This puts much emphasis on the knowledge acquisition phase when developing the knowledge base. As a result a RBR system is useful when the knowledge is well restricted and stable. When common sense is critical, a RBR system is not practical since such knowledge is hard to formalize.

3.3.3 Knowledge representation

As has been mentioned before, the rule is the most common format for Expert Systems to represent the domain knowledge. Within such rule three basic elements can be distinguished [Gevarter, 1987]:

- the ’object description’: the declarative knowledge such as facts which are described in the antecedent- and consequent-part of a rule;
- the ’actions’: the procedural knowledge which describes how to change a situation or to modify the data-base;
- the ’certainties’: some systems allow reasoning with validity or reliability, for which these labels are preserved.

Other common variants on the rule format include semantic nets and frames [Giarratano et al., 1989]. Semantic nets contain propositional information, i.e. statements which can either be true or false. As such the information represented by semantic nets can not be as rich as represented by rules. The structure of a semantic net is shown graphically in terms of nodes and arcs which connect the nodes. The nodes represent the ’object description’, as mentioned in the previous paragraph, and the arcs represent the relations between them. When a semantic net of a family is considered, for example, the arcs represent relationships such as ’is_mother_of’, ’is_brother_of’, etc., while the nodes represent the individual family members. A semantic net provides a convenient way to visualize relationships and can easily be used for inference. However, its expressive power is limited, especially when it gets too large.

While the semantic net is basically a two-dimensional representation of knowledge, a frame adds a third dimension by allowing the nodes to have structures. These structures can be simple values, rules or other frames. Frames can be used to represent typical objects, i.e. stereotypes, by assigning default values to some slots as well as by making use of inheritance. This is a basic characteristic of frames: that it represents related knowledge about a narrow subject which has much default knowledge [Giarratano & Riley, 1989]. Mechanical objects such as cars are well suited for this kind of representation, with its standard components such as the engine, brakes, wheels, etc.. Frames have a higher expressive power and can also be considered more general than simple rule-based knowledge representations. However, when there are too many exceptions of the stereotypes, frames are less effective.

3.3.4 Inference engine

The purpose of the inference engine is to *recognize* rules which seem applicable for the set of facts, and when the *selection* task is automated the inference engine also decides which rule to execute or fire. The execution of a rule, i.e. the *action* task, is usually also controlled by the inference engine, involving constraint solving techniques.

To recognize rules effectively several search strategies have been developed [Steels, 1992]. The ’depth-first’ and the ’breadth-first’ strategies are blind search strategies. Figure 3.11 visualizes these strategies for a goal-driven or backward reasoning process. The depth-first strategy tries to arrive at an end-hypothesis as quickly as possible, moving directly onwards to the next rule when a hypothesis is accepted. With the breadth-first strategy first every possible hypothesis is considered before moving to the next rules.

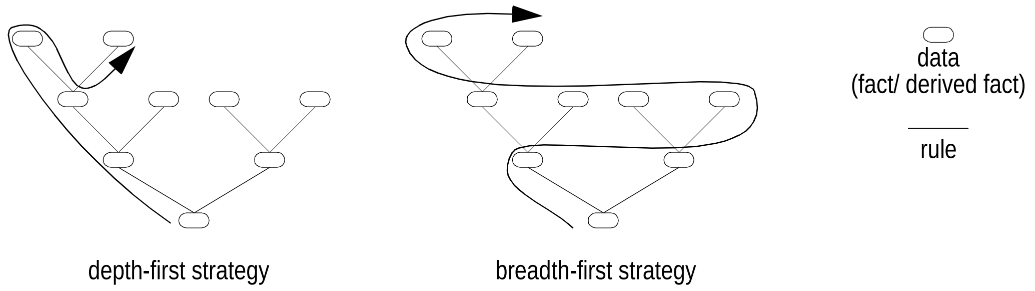


Figure 3.11. Depth-first and breadth-first search strategies.

These blind search strategies only use the syntax of the rules and the data to suggest hypotheses. With so-called heuristic search strategies also domain knowledge is applied. The ’hill-climbing’ method is an alternative of the depth-first strategy. It uses an evaluation function to determine which end-hypothesis is most promising. The ’beam search’ method is an alternative of the breadth-first strategy. Only the best of all possible hypotheses are considered before moving to the next rules. Defining criteria for the determination of the best hypothesis can be rather complicated, though.

As has been mentioned before, RBR is suitable for problem domains which are well defined, well formalised and stable. However, research is carried out to use RBR in combination with fuzzy logic to cope with uncertainties and probabilities [Steels, 1992] [Gevarter, 1987].

Another task the inference engine usually controls, is the action task which leads to new facts. This involves the solving of the relations which are represented by the rules. The relations may be solved one-by-one, at the moment they are fired, or solved as one set when the network of rules has been completed. When the network of rules has been generated, the solver is called upon every time the input parameter values of the network are changed. A changed value can make a rule invalid. In that case the inference engine rejects the rule and has to search for other, valid rules to restore the network.

3.3.5 The *action* task: solving the relations and constraints

When the rules in RBR are expressed as constraints, the *action* task in the RBR cycle has to find a solution that satisfies the complete set of constraints. If no appropriate solution can be found, the constraint problem is called over-constrained When multiple parameter value ranges are possible the problem is under-constrained, in which case an optimization function may be applied to find the preferred solution.

The problems involved in solving the set of constraints depend on the type of constraints. Next several characteristics are mentioned.

One characteristic is the direction of the constraints. A constraint can be unidirectional, bidirectional or multidirectional. In a unidirectional constraint one variable is determined by the other variable(s), i.e. there is a master-slave relation between the variables.

Another issue which may complicate the constraint solving is that the set of constraints does not have to represent a hierarchical tree. This can be caused by two unidirectional constraints which direct the same variable, i.e. when two master-slave relations share their slave-variable. A change of one master-variable may result in a modification of the slave-variable which is conflicting with the constraint with the other master-variable. The complexity of constraint solving methods is also increased when constraints are bidirectional (or multidirectional when more then two variables are involved), implying that there is no master-slave relation between the involved variables. This may lead to ambiguities or several appropriate variable domains that satisfy the constraint set, especially when multidirectional constraints are involved.

Another complication arises when three or more (unidirectional) constraints generate a loop between the variables, i.e. variable *a* depends on variable *b* which depends on variable *c* which depends on variable *a*.

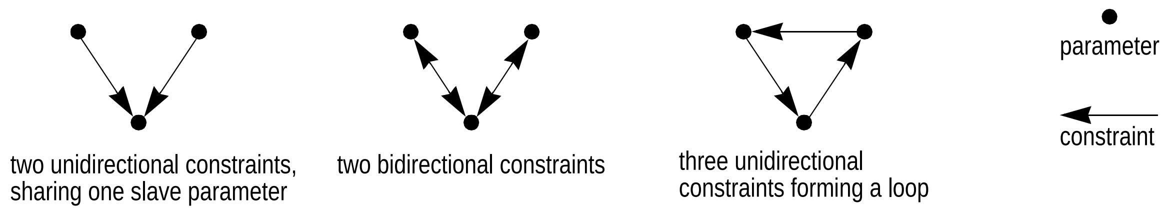

two unidirectional constraints, sharing one slave parameter
two bidirectional constraints
three unidirectional constraints forming a loop
parameter
constraint

Figure 3.12. Examples of constraint combinations resulting in a non-hierarchical structure.

Many constraint solving techniques have been developed. In [Dohmen, 1995] a comprehensive summary is presented of the most common, general constraint solving techniques. The two main types of solving techniques are mentioned: simultaneously solving techniques and local propagation techniques.

In simultaneously solving the constraints are usually expressed by algebraic relations, which are solved simultaneously with numerical methods. In these techniques it is iteratively tried to find approximations of the variable values, using initial starting values and a convergence criterion. Well-known examples are the relaxation methods, the Newton-Raphson iterations and the Simplex technique. These solving techniques are able to deal with loops of constraints.

In local propagation techniques the constraints are explored one by one, from one variable to another. Searching techniques can be employed to determine a suitable path in the constraint network, i.e. the ordering of the constraints, and to find the appropriate variable domains or values. Loops can usually not be handled by local propagation techniques.

Before one of these two main solving techniques are applied, one can try to simplify the constraint solving problem. By symbolic algebra techniques the set of constraints may be formalized into another set which is expected to be easier to solve. Or the constraint set can be divided into more or less independent sets, to be solved by either of the techniques.

3.3.6 The role of RBR in design

As has been mentioned in the introduction of this chapter, the *if-then*-scheme of Rule-Based Reasoning seems to be applicable for every step in the design cycle of Figure 1.1: to suggest a new artefact or refinements, to predict its performances, to evaluate these predictions and to propose modifications.

However, this chapter has also revealed the limitations of RBR. For practical applications, the problem domain should be restricted and well defined, and the domain knowledge should be well formalised and stable.

These critical notes reduce the applicability of RBR in design considerably. In Section 3.2.9 the three main categories of design have been mentioned: routine design, innovative design and creative design.

In the area of creative design RBR does not seem to be usable due to the open character of the problem domain. An open domain is essential for creativity, but is impossible to be covered by a knowledge base. Only the evaluation task of the design process, i.e. comparing the required with the predicted performances, may be supported with RBR techniques.

In innovative design the problem domain is known, but the problem-solving strategy is not.This makes it still hard to use RBR techniques for generating data, such as suggesting a new artefact or refinements and proposing modifications. But it seems possible to support the data evaluation step and to predict the performances. The issues of knowledge acquisition and knowledge formalisation are critical, though. To be of practical use the problem domains should still be as small as possible.

On the other end of the spectrum is routine design. All required knowledge is known in advance, hence even the first phase in the design process, suggesting a new artefact seems possible to be supported with Rbr techniques. In general these steps are based on simple heuristics and rules-of-thumb. The issue of knowledge acquisition should not be underestimated, though.

3.4 Geometric representation

3.4.1 Introduction

A geometrical model describes the 3D geometry of an object. When a geometric description has been generated, it can be visualized, hence providing some form of feedback to the designer of the object. From a geometrical model also physical properties can be deduced, such as the volume, weight, centre of gravity, etc.. Furthermore a geometry can give some insight in the manufacturing of the object. So it is clear that geometry modelling is essential in the design process.

Therefore much effort has been spent on the area of geometric modelling. Several techniques have been developed, such as surface and solid modelling. As the name suggests, surface modelling focuses on the shape of surfaces. A 3D object can be presented by a set of surfaces. The validity of the resulting 3D object is the designers responsibility. This includes preventing gaps between surfaces, keeping track on which side of the surface the object extents, etc.. In solid modelling the modeller takes care of these issues and offers solid modelling operations that affect the 3D shape directly, such as union, difference and intersection. Therefore a solid modeller is much more complex than a surface modeller. In Section 3.4.2 both techniques are reviewed.

To be able to change a geometric model relatively easily, the technique of parameterised modelling can be applied. In this technique, parameters are used to specify and manipulate the topology and geometry of a model. These parameter values can easily be varied, resulting in an altered topology (for instance altering the number of holes in an object) or a changed geometry (i.e. changed dimensions such as the length and width of an object). Some changes may involve a new interpretation or construction of the shape definition. Section 3.4.3 examines this technique further.

To offer more flexibility, constraint modelling techniques have been developed. With constraints the geometry can directly be defined by specifying relations between geometric elements, such as parallelism and distances between faces. Also higher level parameters can be used, such as the volume of an object. Constraints are specified in a declarative way, which requires a separate solver to keep all constraints satisfied, i.e. maintain the validity of the model. Issues involved in the constraint satisfaction are discussed in Section 3.4.4.

The last decade shows the development of feature-based modelling techniques, to give the designer even more support. Features can represent geometric entities such as holes, slots, roundings, etc.. On top of that features also contain the functional meaning, engineering significance or context of such an entity. With features the syntax of a geometric entity is extended with semantics. For example the geometric entity "hole" only has a meaning if it is completely surrounded by material. When a hole is defined in an object, the feature-based modeller checks for this condition. But features can contain much more than geometric information only; their purpose is to capture the designers intent [Shah & Mäntylä, 1995].

Because of this feature-based modellers are not considered in this thesis. Instead, some words are spent on the discussion of the geometric modelling techniques and the (general) constraint-based modelling techniques. Together with a brief summary this is described in the last section about geometric representation, Section 3.4.5.

3.4.2 Methods to specify geometry

As has already been mentioned in the previous section, two methods can be distinguished to specify a 3D object: surface modelling and solid modelling. In this section some technical details about both types of modellers are summarized. In addition alternative representation techniques are mentioned which have been developed to support the user input. Internally these alternative representations are transformed into one of the two model types.

Surface modelling

In surface modelling the surfaces are defined by mathematical descriptions. Several surface shapes are possible, resulting from different descriptions:

- linear surfaces, such as polygons;
- quadratic surfaces, such as cylinders and spheres (usually implicit equations);
- free-form surfaces, such as Bézier, B-spline and NURBS (Non-uniform rational B-spline) surfaces (parametric representation).

Bézier, B-spline and NURBS surfaces can be defined with the aid of supporting points. By using these points the user is able to define complex curved surfaces without having to deal with the underlying mathematics. These mathematics show a parametric representation instead of implicit equations, which offer more flexibility with regard to translations and deformations [Farin, 1990].

A surface can also be split up into smaller surfaces, called patches. Large, curved surfaces can exactly be modelled with free-form patches, or well approximated with simpler linear patches.

Solid modelling

A solid modeller cares for the validity of the constructed 3D objects, and offers typical "solid modelling" operations such as union, difference and intersection, but also blending operations such as rounding of edges. Two types of solid representation techniques can be distinguished [Bronsvoort & Post, 1993]:

- Boundary representation (B-rep)
  B-rep defines a solid by its boundaries, i.e. its boundary surfaces. The surfaces can be defined as described at "Surface modelling" in the previous paragraph. In a B-rep such surfaces are called *faces,* which are bounded by *edges* on each side, which on their turn are bounded by *vertices* on both ends. A topological data structure is maintained, representing the relations between these elements. Figure 3.4 makes this clear.

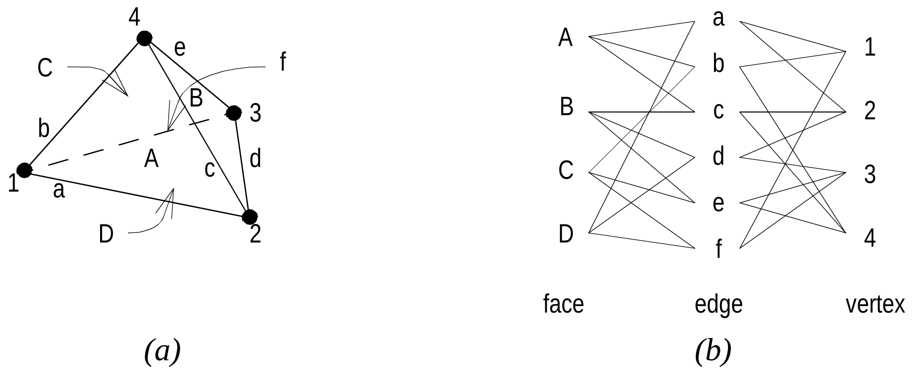


Figure 3.13. Boundary representation of (a) a quad-surface (4 faces); and (b) its topology.

- Constructive Solid Geometry (CSG)
  With CSG, an object is constructed from primitives such as blocks, cylinders and spheres. The general way to define these primitives is by so-called half-spaces. A half-space splits the space into two by a surface and with material on one side. The half-space is defined by the mathematical description of the surface, usually linear or quadratic, and an indication which half of the space is implied. A primitive is defined by the intersection of several half-spaces. Figure 3.5 shows the definition of a cylinder by three half-spaces.

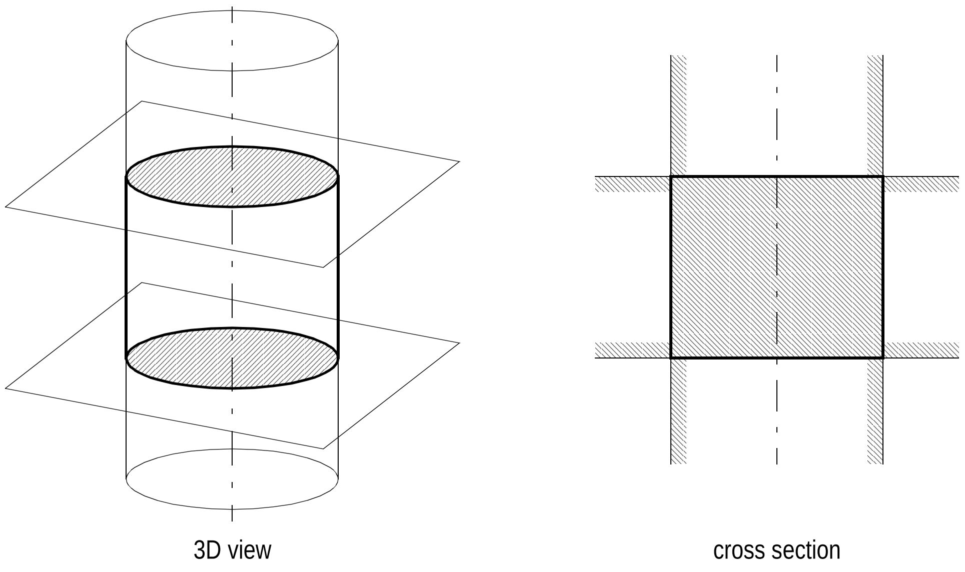


Figure 3.14. Representation of a CSG primitive (a cylinder) by half-spaces.

Additional input representations

To support the user, alternative input representations have been developed. Internally, however, the modeller will use its surface or solid representation.

- input by a cloud of many 3D points
  Such input is for example generated by measuring existing free-form shapes. By interpolation of the measured points and smoothening techniques it is tried to deduce the original, existing surface.
- input by 2D drawings
  A 2D-3D conversion is for example required to generate a 3D model from 2D technical drawings. Because explicit information about faces, edges and vertices is usually not available, this may result in multiple interpretations of the 2D drawings. Therefore interaction with the user is usually required.
- input by procedures
  A shape can be defined indirectly by a procedure instead of directly, which requires only a small amount of geometric data. Typically the methods of cross-sectional design belongs to this category. Sweeping is a good example of such a method. With sweeping a shape is generated by moving a 2D closed curve (cross section) along a trajectory which is a 3D open or closed curve. Validity problems arise when the shape intersects with itself.
- modification by shape operations
  An initial model can be modified or extended to add details. Examples are the smoothening of discontinuous transitions between surfaces and rounding edges, which are called blending operations. Other types of shape operations can be used to compose complex shapes from elementary shapes, without changing its topology. Bending and twisting a geometry are such operations.

3.4.3 Parameterised geometry

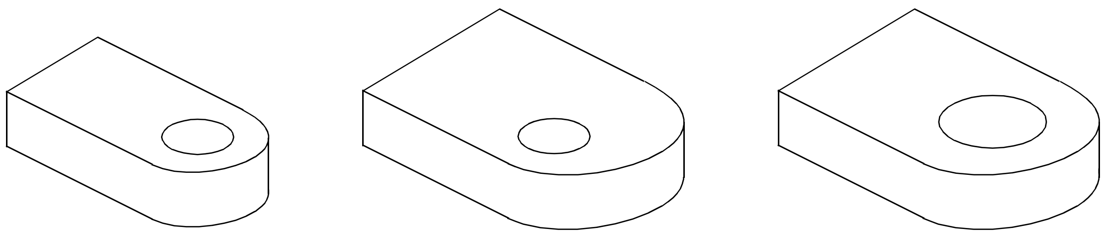
To facilitate the modifying of a model, its dimensions can be associated with parameters. The length and width for example can easily be varied by changing the parameter values, instead of changing the values inside the model definition. In addition it is possible to define relations between parameters.

Also the construction of models is made easier by using parameters. Geometric primitives are assembled to compose standard entities which represent objects or parts of objects, such as bolts and screws or specially shaped holes (this is in close resemblance with feature-based modelling). These parameterised entities are collected into a library. When constructing a new model, the user can select a parameterised entity and provide it with particular parameter values. This particular entity in an object is called an instance; the parameterised entity in the library is called the generic entity. If the generic entity is modified afterwards, the instances will change accordingly.

Generally, three types of parameters can be distinguished:

- position and orientation parameters
  An entity has six degrees of freedom (dofs): three dofs referring to the position in the 3D space and three dofs referring to its orientation. Hence with six parameters the position and orientation of an instance are defined.
- geometrical parameters
  The geometrical parameters define the dimensions of the entity, such as lengths, widths and diameters.
- topological parameters
  With topological parameters a pattern can be defined, such as the repetition of instances in an object. A pattern is parameterised by one or more integer variables that control the number of instances in the pattern, and one or more real variables that specify the distances between them.

One of the major benefits of parameterised modelling is that the parameters of one instance can be related to those of another instance. This is very useful when composing parts from several entities. This offers a flexibility which is explained in Figure 3.15.



(a) object composed of two entities: "rounded block" and "hole".

(b) changed "rounded block" dimensions, independent "hole".

(c) changed "rounded block" dimensions, dependent "hole".

Figure 3.15. Effects of relations between parameters of several entities.

Figure 3.15 shows that different kinds of parameter types can be related to each other, in this example geometrical as well as position parameters. The relations introduce a master-slave dependency between parameters: in Figure 3.15.(c) the position parameter of the hole is dependent of the width parameter of the rounded block. In its simplest form all these relations together can form an hierarchical tree (e.g. a scene graph) where modifications of the root parameter will affect the parameters at the leaves, but not the other way around. The modifications are propagated in a unidirectional way. Hence the designer should already have an idea of possible modifications when defining the relations between parameters.

Although the user does not necessarily make a distinction between the different types of parameters, the modeller does. When topological parameters are modified, the modeller has to redefine the geometry, i.e. generate a new topological data structure. Modifications of dimensional parameters do not require a new set-up of the topological data structure, although the structure may have to be changed due to for example new or lost intersections of surfaces.

3.4.4 Constraint-based geometry

A constraint defines a relation between parameters that should be satisfied. For geometric applications, constraint-based modelling can be regarded as an extension to parameterised modelling. The main difference is that the maintenance of the constraints is performed separately, by a constraint solver. Constraints are expressed in a (more) declarative way, which makes it also possible to deal with for example non-hierarchical constraint structures, as opposed to the simple hierarchical trees as mentioned in the previous section

Several types of constraints exist, representing different types of relations. There are constraints with an algebraic character, which are for example used to express physical relation between the stiffness of an object and its dimensions.

Geometrical constraints form a special kind of constraints. Typically they involve the position and orientation of entities with respect to each other, such as parallel surfaces and touching of entities. In addition geometric modellers have often implemented efficient algorithms for the determination of geometrical parameters, such as the volume and the centre of gravity of objects. Such parameters can be used in the (geometrical) constraints.

The issues involved in satisfying and maintaining the constraint sets are referred to as constraint satisfaction problem (CSP) issues [Dohmen, 1995]. In Section 3.3.5 some of the main characteristics of CSP have already been mentioned, in the context of Rule-Based Reasoning. Identical issues play a role in Geometric Modelling. In addition specific geometric constraint solving techniques have been developed. The implementation of these dedicated techniques differ much from the more numerical oriented constraint solving techniques.

Another term which is often used in the framework of constraint solving and parameterised modelling is variational modelling. In general, variational modelling seems to be more related to CSP than with parameterised modelling. Because of the many different interpretations of the term, and the minor significance of these differences to this thesis, the term variational modelling will not be used here.

3.4.5 Discussion

The previous sections have shown that many methods are available to model a geometry. They range from explicit specifications which do not offer much flexibility, to the higher-level specifications with declarative constraints. The higher flexibility is usually gained at the cost of increased overhead work. The declarative constraints, for example, require a separate solver to maintain the validity of the geometric model. Because of this flexibility, constraint-based modelling techniques can be very useful for supporting the design process.

As has been mentioned in Section 3.4.4, algebraic relations can be implemented by constraints, such as engineering (functional) relations, but also geometric relations such as parallel surfaces. In the ideal design environment both constraint types should be satisfied by one solver. However, the implementation differences between physical constraint solvers and geometric constraint solvers are (still) too large. A combined solver is currently not available.

A practical alternative is to integrate the two different solvers as much as possible. Therefore the communication between both solvers should be facilitated; see [Dohmen, 1995].

When reflecting Section 3.3, it is obvious that there is much resemblance between (general) constraint-based modelling and the Rule-Based Reasoning (RBR) techniques: the *if-then*-rules also represent (engineering) relations in a declarative way. The RBR techniques focus more on the generation of the network of relations, though, whereas constraint solving techniques focus on the solving of the constraint set, i.e. the network.

4 Concept of the AIDA system

4.1 Introduction

The goal of the AIDA-tool is to support the designer in the conceptual design of aircraft: one or more feasible aircraft concepts are generated which are supposed to satisfy the design specifications. A concept describes the aircraft configuration and its primary dimensions and characteristics.

The background information which lead to the set-up has been described in the previous chapters.

In Chapter 1, design theory has been discussed. It is argued that the design process is a non-deterministic process: a one-to-one relation between the design problem (performance specifications) and the design solution (artefact) does not exist. That is, the artefact can not inevitably be deduced from the specified performances. Instead the design process is an iteration process, continuously running through the design circle of Figure 1.1: suggesting artefacts and/or refinements, predicting its performances, evaluating the performances, proposing modifications, etc..

In Chapter 2, the characteristics of the conceptual design of aircraft are described. It described the kind of design decisions which are taken, such as discrete decisions about the aircraft configuration and continuous decisions about the magnitude of dimensions. Also the kind of arguments on which they are based have been examined. Because there is no direct link between the aircraft configuration only and the functional performances, many decisions are based on experience. The experience is available by means of empirical relations, statistical tables and existing designs.

Once an aircraft configuration is suggested together with some primary parameters such as the weight and wing shape, some explicit, simple relations become accessible to perform a first prediction of the aircraft performances. Good knowledge about the reliability and accuracy of those relations is required to evaluate these predictions. Then the designer is capable to close the first design cycle by proposing proper modifications.

In Chapter 3, some Artificial Intelligence techniques have been examined which are able to support these specific design tasks.

Based on the first three chapters, the next aspects within the design process are regarded to be suitable for support by the AIDA system:

- generating initial aircraft concepts which form a reasonable starting point for modifications;
- performancing parameter studies in order to propose modifications of the initial concept;

- modelling the concept to visualize it for a better understanding of the designed concept.

In the AIDA system the supporting activities are implemented in separate modules. Hence the three above mentioned aspects are supported by three modules:
- a Case-based reasoning (CBR) module, to generate an initial concept
  This module is implemented in an existing tool, developed by Netten as part of the EADOCS project [Netten, 1997]. It is described in Section 4.3.
- a Functional module, to predict the performances of the concept
  This module is also implemented in an existing tool: QUAESTOR, which has been developed by Van Hees [Van Hees, 1997] using Rule-based reasoning (RBR) techniques. Section 4.4 describes this module.
- a Geometrical module, to generate a 3-D model from the concept data
  This module is implemented with the commercial software package Pro/Engineer [PTC, 1999], a feature-based modeller. The implementation is described in Section 4.5.

To clarify the roles of these three modules, they can be placed in the design cycle of Figure 1.1 according to Figure 4.1. With the CBR module a case (or a combination of cases) is suggested which contains basic data about an artefact. This basic data is enough for a first set of predictions supported by the Functional module. The Geometrical module transforms the basic case data into a 3-dimensional model, which is used for a visual evaluation. Together these three modules help to complete the data of an artefact, comprising structure, behaviour and function data.

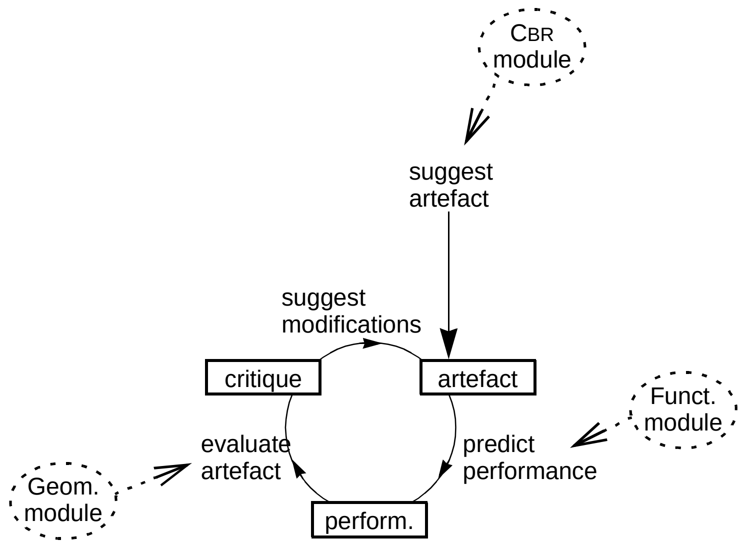


Figure 4.1. A rough indication of the supporting roles of the different AIDA modules within the design cycle.

The next section elaborates more about the general set-up of the AIDA-project.

4.2 General set-up

In Chapter 2 the conceptual design of aircraft has been analysed and several design tasks have been distinguished. These tasks are suitable to be implemented by different AI techniques, which have been described in Chapter 3. In the AIDA-tool each technique has been implemented in a separate module. Four modules have been developed:

- Case-based reasoning (CBR) module
  In this module CBR-techniques are applied to generate an acceptable initial aircraft concept, which can be modified to get a feasible aircraft concept.
- Functional module
  This module uses RBR-techniques to perform sensitivity studies on the primary parameters of the aircraft concept. With these studies the initial concept is modified into a feasible aircraft concept.
- Geometrical module
  This module automatically generates a solid model of the concept, using feature-based techniques.
- Central user interface (CUI)
  This module handles the data communication between the modules.

A scheme of how these modules are connected to each other is sketched in Figure 4.2.

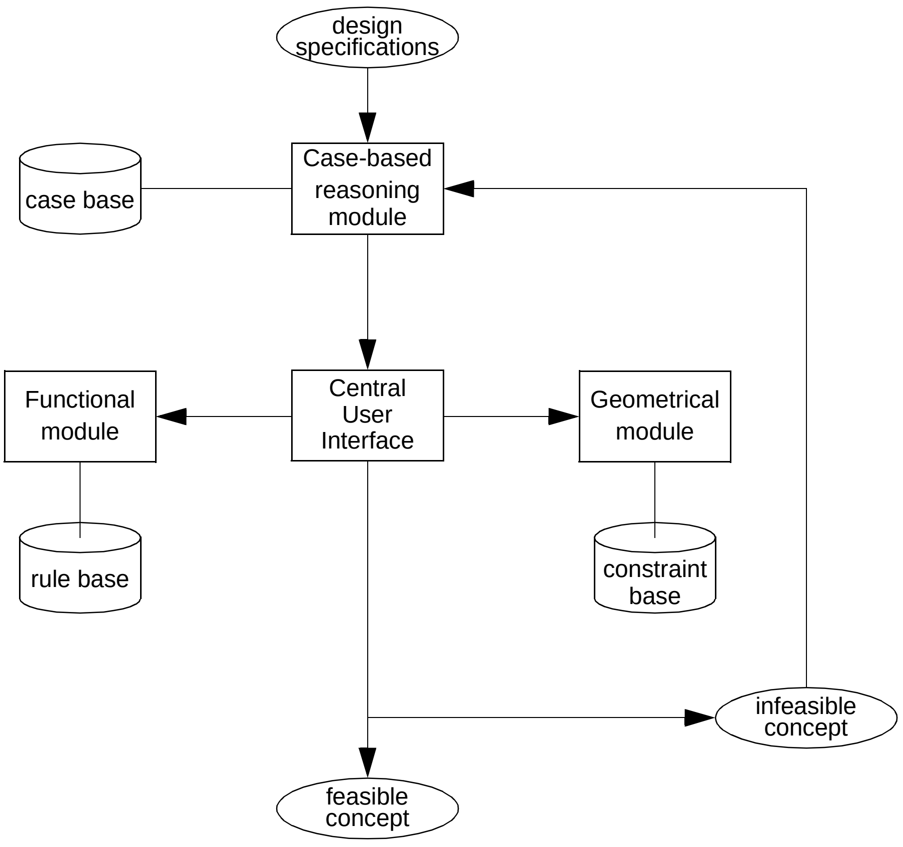


Figure 4.2. Modular set-up of the AIDA-tool

First, the user defines the design specifications. The specifications vary from design problem to design problem. They usually consist of performance specifications, such as the minimum range with a certain payload or the minimum cruising speed, and/or topology specifications, such as a specific type of engine, and/or geometrical specifications, such as the maximum span (because of airfield category).

With the Case-based reasoning module the experience of previous aircraft designs is used to make some initial design decisions. The module searches for existing aircraft in the case-base which resemble the design specifications. It rearranges the order of the cases according to their similarity, and the users selects one of them.

In addition to the selection of a case, the Case-based reasoning module is also used to adapt the selected case. That is, to mix data of more cases in order to improve the similarity with the design specifications. At this design phase the focus is on the discrete data, i.e. the topology, such as the number of engines, the type of engines and the type of tail.

The implementation details of the Case-based reasoning module is described in Section 4.3.

The selected (and adapted) case serves as the starting point for rest of the design process. Important topological choices have been made, and some continuous parameters have been given an initial value. The provided level of detail of the data enables the use of some basic empirical relations. With these relations the solution space can be explored, so that appropriate modifications on the selected case can be applied.

The Functional module supports the designer with these modifications. Using rule-based reasoning techniques, the module provides a flexible way to apply the appropriate empirical relations for a wide variety of design problems. A network of these relations is built interactively with the designer.

This module is implemented with QUAESTOR [vanHees,1997], a numerically oriented expert system which has been developed for conceptual ship design. More details are described in Section 4.4.

An important aspect of the AIDA-tool is the visualization of the aircraft concept. This is handled by the Geometrical module. The module automatically generates a geometric model of the aircraft, using pre-defined standard elements. The set of pre-defined aircraft parts determine which topologies can be visualised. The variation of continuous dimensions is limited by geometrical constraints.

The module is implemented in the feature-based solid modeller PRO/ENGINEER [PTC,1999]. The module is described in Section 4.5.

A practical issue is the communication between the modules and between the modules and the user. Because the described modules are implementations of independently developed tools, they use different data-formats. The Central User

Interface module takes care of this problem by leading all communication via this module.

Also, some user-interface requirements are not supported by the individual modules. For example the presentation of specific parameter studies can not be handled by the Functional module. These extra presentation tasks are incorporated by the Central User-Interface module.

The details of this module are described in Section 4.6. This module was developed in VisualC [Schipperijn, 1998].

4.3 Case-based reasoning module

4.3.1 Tasks

The CBR module supports the generation of an initial aircraft concept. This initial concept is used as an acceptable starting point for the modifications by the Functional module, and contains the information required to build the solid model in the Geometrical module.

This initial concept is considered acceptable when it is thought to be (more or less) easily modified to fulfil all design specifications. Therefore this process is guided by the set of specifications.

The generation of the initial concept involves choosing an appropriate aircraft configuration, as well as making appropriate estimations of some primary dimensions. Hence this concerns the quantification of discrete as well as continuous parameters.

4.3.2 Implementation of case-based reasoning technique

Case-based reasoning techniques are used to generate the initial aircraft concept. This technique has been chosen because its characteristics as are mentioned in Chapter 3.

An important characteristic is that each case in the case-base embodies a direct link between Function, Behaviour and Structure data. Using these links enables it to apply the relations between the data, without knowing them explicitly. With this black-box approach it is possible to propose an initial aircraft concept based on the design specifications only, avoiding the complex evaluation of all arguments. It is a technique to reuse the design experience which is implicitly contained in existing aircraft.

Another aspect of CBR is that new experiences can be added in the form of new cases. This can be considered as a way of learning.

The strategy used for the CBR module is to first retrieve a ’best-matching’ case completely, and, when needed, to reuse parts of other cases for adaptations. Adaptations are particularly intended to change the configuration, i.e. to modify the discrete parameters. The adaptation enables the combination of all kinds of aircraft configurations which are collected in the case-base.

Combining parts of cases introduces inconsistencies within the adapted case, however. That is, the relations between the data of the adapted case become invalid. Hence the adapted case has to be evaluated first before next adaptations can be usefully applied.

The CBR module used for AIDA has been developed by Netten as part of the EADOCS project [Netten, 1997]. This tool supports the conceptual design of fibre reinforced composite sandwich panels, using constraint-, case- and rule-based reasoning. Only the component which implements the case-based reasoning technique has been used for the AIDA project.

The CBR module maintains a database with cases, the case-base. Each case is a description of the Function, Behaviour and Structure of an aircraft. The aim of the CBR module is to support the choice of a case from the case-base that best matches a set of specifications put forward by the designer. These design specifications in general relate to the Function parameters, but may also describe Behaviour and Structure parameters. This set of design specifications is collected in the so-called target-file. The parameters used in the target-file (partly or completely) correspond with the parameters in the case-files. The similarity between a case and the target is determined by the number of matching parameters and their values of both files.

A kind of qualitative matching is applied by using parameter domains instead of their exact values. For each parameter a number of domains is defined, which discretisizes the global range of its values. For example a parameter p_2 which can only be positive, can be linked with the domains [0,0.1), [0.1,0.5), [0.5,1.0) and [1.0, ∞). These domains may be associated with a ’small’, ’moderate’, ’large’ and ’extreme large’ p_2. A case has a similarity for a certain case parameter when the parameter value is within the same domain as the user-specified parameter value in the target-file. Figure 4.3 shows the matching between target parameter $p_2=0.8$ and the case parameter $p_2=0.7$. For this case there is no matching for parameter p_1 due to different domains, and no matching for parameter p_3 because the case does not contain that parameter.

The number of parameter domains of target and case that match, is a measurement for the similarity between the case and the target. The cases are ranked according to their similarity scores. For example case 3 in Figure 4.4 has a better similarity score than case 2 and case 1, having two matching parameter domains compared with one and zero respectively.

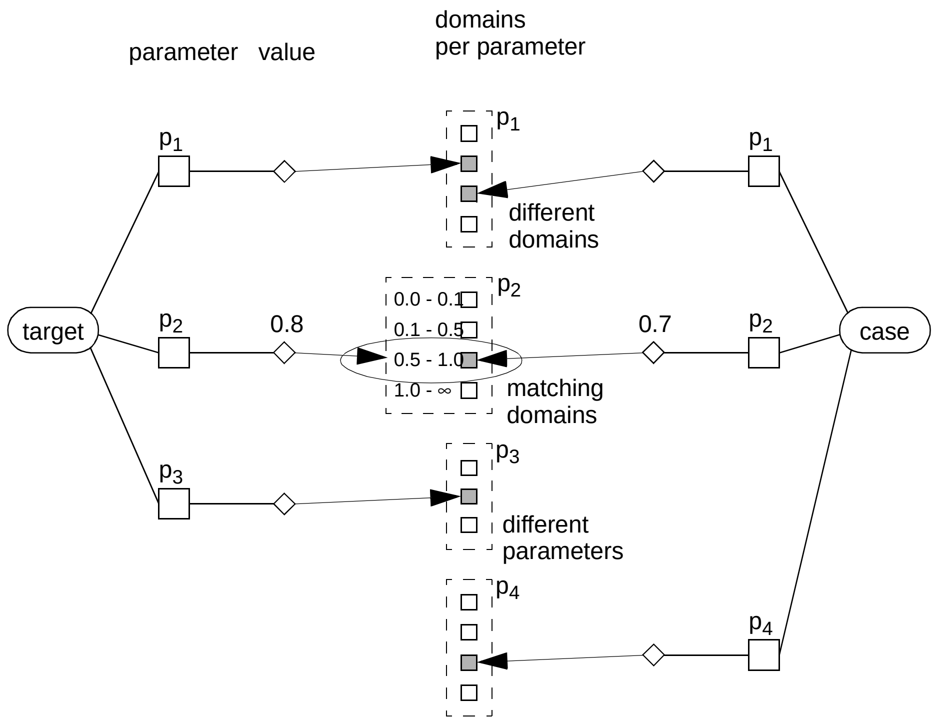


Figure 4.3. A similarity between a target set and a case, based on similar parameters and similar parameter domains.

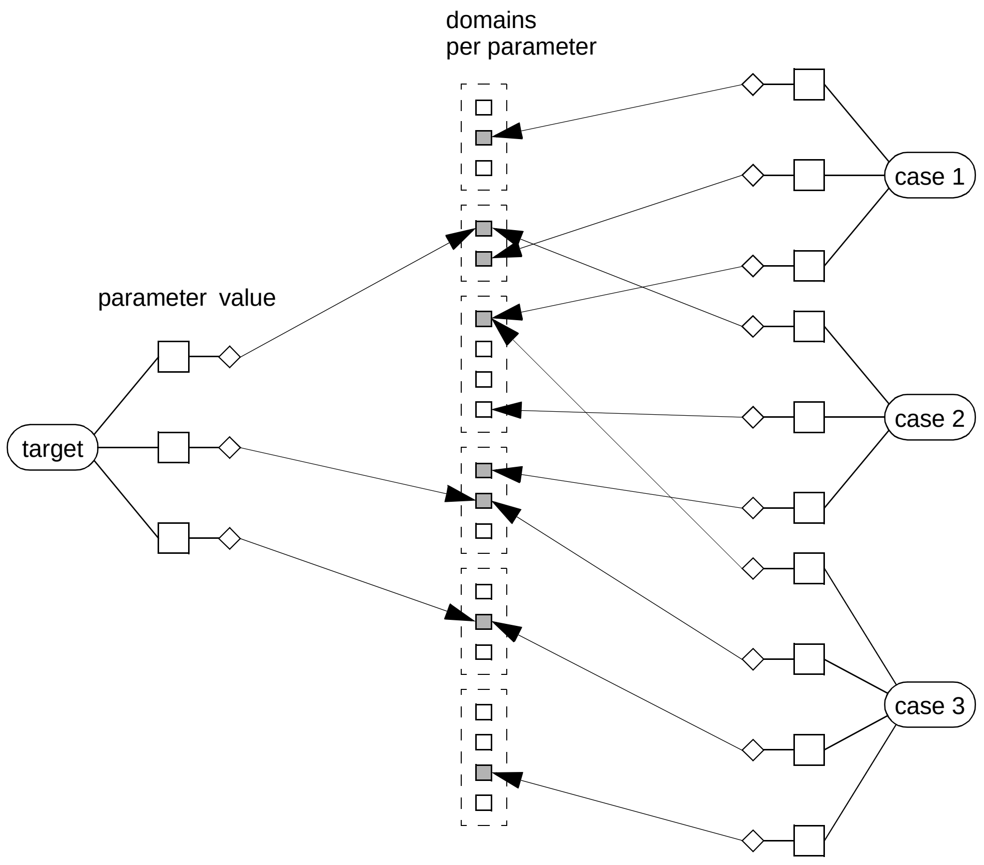


Figure 4.4. Different similarity scores between the target and three cases.

The designer generally chooses the case with the highest score, since this case is assumed to be most promising in satisfying the design specifications. However, it may also be advantageous to choose a case with a lower score but which seems to provide more flexibility towards modifications in order to satisfy the design requirements.

Note that the case-files may contain different sets of parameters. This is because the description of different configurations can involve different parameters, or because some parameter values of the cases may be unknown to the developer of the case-base.

Another aspect which is shown by Figure 4.4 is that the parameter set of the target-file does not have to correspond with the parameter sets of the case-files. It is even reasonable that the target-file consists of fewer parameters than the cases, because the purpose of the CBR-module is to retrieve a complete case from the case-base, including Function, Behaviour and Structure parameters, while the target-file contains the design specifications which in general consists of Function parameters only.

The process of choosing a case that best matches the target specifications contains the following steps:

- collecting the case-data
  First the data of cases have to be collected and put into case-files. It is important to define the case-parameters unambiguously to be able to compare the data.
- defining the parameter domains
  Per parameter the global range of values is discretisized into a number of domains.
- indexing the case-files with respect to the parameter domains via the case-data
  Based on its parameter values, each case-file is linked with a number of parameter domains. First the relevant parameters are identified, then the fitting domains are labelled. The process of identifying the relevant parameters is assisted by a hierarchical classification structure of the parameters.
- indexing the target data with respect to the parameter domains
  Like the previous step, the target-file is associated with a number of parameter domains. The target data is organized in the same classification structure as the case-data, for the same reasons. In addition each target object has been given a priority number to be able to express different importances of the matchings.
- measuring the similarities between the target set and the cases
  The similarity score of a case depends on the number of (parameter domain) matchings between the case and the target set. These matchings are identified by the links between the target file and the parameter domains, and between the case-files and the domains.

The similarity score is expressed by a number between 0% and 100%. When a case has a similarity of 100%, each parameter value of the target set matches with a parameter value of the case; i.e. share the same parameter domain.

The similarity score also depends on the classification structure of the parameters. First the score is determined per target object, that is per group of parameters which belong to one class according to the classification structure. This score determines the so-called local similarity. The global similarity score of a case is the weighted average of all local similarity scores of that case. The weight of a target object is set by the priority number, to express the importance of the group of parameters.

- ranking the cases according to their similarities
  The matching results are shown to the designer by a list of cases with their global and local similarity scores, i.e. with their similarity with the total target set and with the target objects, respectively. The weighted average of the local scores directs the global score of a case.
- select a case, usually the best-matching case
  The designer selects a case. Usually this is the case with the highest global score, but the designer may choose another case which offers more flexibility for adaptations and modifications in next design steps.

Usually the designer adjusts the target-file several times in order to better understand the influence of the different target parameters and the priority numbers.

To speed up the matching process, some pre-processing work is done before the CBR-module is applied. This work consists of the creation of an indexing network between the case files and the parameter domains. The network is used to efficiently count the number of matchings between each parameter domain and the case-objects and the cases. The indexing mechanism is based on the classification structure and the spreading of the case-data among all the parameter domains, such as depicted in Figure 4.4. Figure 4.5 shows the position of the indexing network. The objects are the instantiations of classes from the classification structure.

When the indexing network has been created, the designer defines the target file. The CBR-module searches for matching parameters and domains, and determines the similarity scores per case and per case-object.

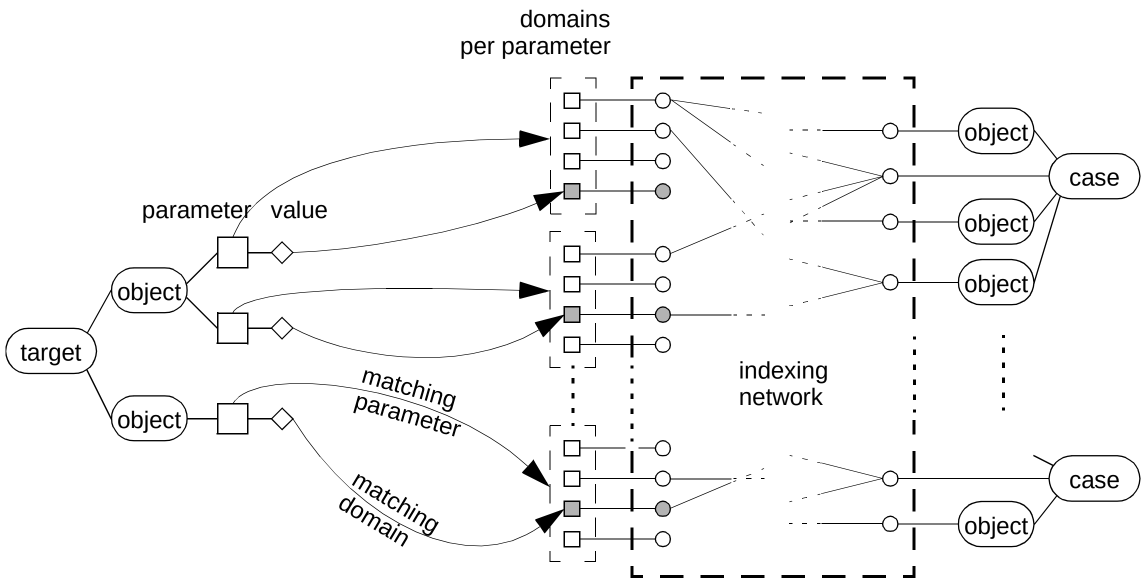


Figure 4.5. Linking the target set with cases via the indexing network.

4.3.3 Case representation

Each case contains data which allows to compare aircraft and to define a solid model of the aircraft. The data is structured according to the Function, Behaviour and Structure typology, as is described in Chapter 3.

The aircraft model is described by Structure data; which parameters are used is discussed in Section 4.5. It is tried to avoid dimensional parameters, such as length and width, but to use non-dimensional parameters instead, such as length-width ratio and surface area. The idea is that these parameters have more meaning to the designer because they describe the shape and size of an artefact separately.

However, cases do contain dimensional parameters such as length and width, because they can be subject of comparisons. For example the (maximum) wingspan can be specified. As a result, cases contain a lot of redundant information. The case-base developer should be cautious for inconsistencies between the data.

Behaviour parameters typically describe the qualities of the aircraft and its components, such as the wing-loading (maximum aircraft weight / wing area) and the maximum liftcoefficient. These parameters are very useful to compare aircraft, and are often subject to parameter studies in next design steps. Because many of these parameters are combinations of other parameters in the case, such as the wing-loading depending on the maximum weight of the aircraft and the wing area, the case-base developer should again be cautious to exclude inconsistencies.

Dependent on the design problem, the designer can be interested in any combination of parameters. To be used for similarity measurements, however, these combinations have to be defined in the cases already. So, the developer should know which Behaviour parameters may be important when building the case-base.

The Function data describe the performances of the aircraft, such as range and speed. These parameters are usually specified in the set of design specifications, and will therefore form the main portion of the target set.

It is important to have equal conditions for each Function parameter in order to be able to compare data properly. Other conditions may result in other performances. This can be a problem, because the data sources such as the Jane’s collection [Jane’s] do not always describe the conditions accurately.

Each of these three main data-classes, i.e. Structure, Behaviour and Function, have been decomposed into an object-oriented structure.

The basis of the organization of the Structure data is the decomposition of the aircraft into standard components. The class Structure is directly usable by the Geometrical module, which is described in Section 4.5. Components with a lot in common are classified as specializations of more general classes; for example wings and tail surfaces are specializations of lifting surfaces.

The object-oriented structure of the Behaviour and Function classes is based upon related groups of parameters. That is, all weight parameters are put in a class called Airplane_Weight, all speed parameters are put in a class called Speed, etc.. In general, the purpose of this organization is purely to order all data.

For some parameters there are also other reasons. Those parameters have such a strong influence on each other, that they have to be considered together in the similarity measurements. Examples are the range and payload parameters; without knowing the payload it is not reasonable to compare the maximum range of aircraft. The usual procedure is to define the range parameter in some standard situation, such as at maximum payload or at maximum fuelload. However, when the sources do not refer to these standards, it is useless to consider the range parameter only. The two parameters range and payload have to be considered together, in pairs. This is possible by labelling both parameters to the same class. Only when both payload and range parameters of a case match the target set, the relevant case-object will score 100%.

4.3.4 Adaptation with the CBR-module

Based on the similarity with the target-file, the designer selects a case of the case-base. Usually, this case does not meet all the specifications which are put in the target-file. Parts of another case can be copied onto the initial case to improve the concept. This is called adaptation. Figure 4.6 shows the strategy which is followed in the adaptation procedure.

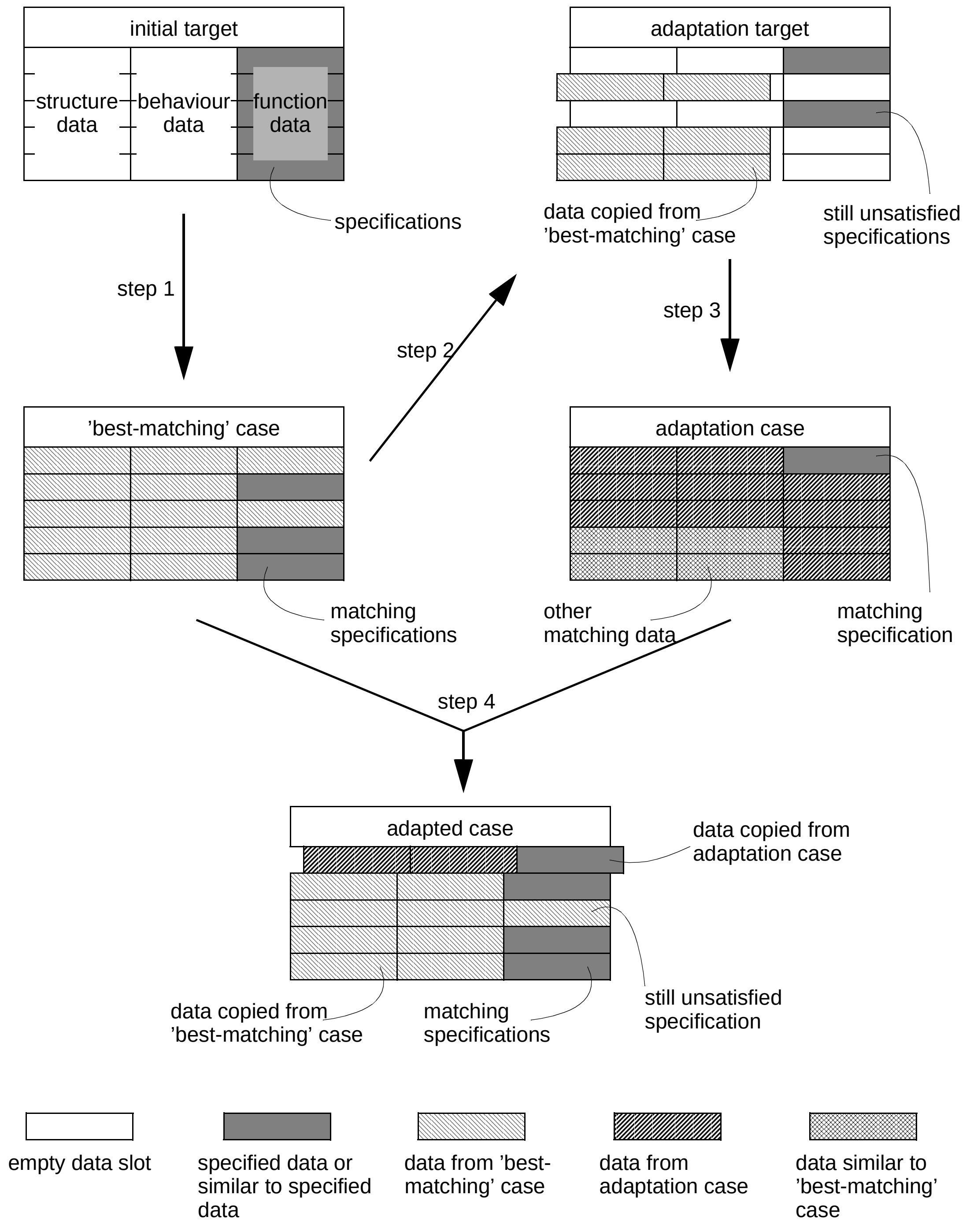

initial target
structure data
behaviour data
function data
specifications
adaptation target
data copied from 'best-matching' case
still unsatisfied specifications
step 1
step 2
step 3
'best-matching' case
matching specifications
adaptation case
other matching data
matching specification
step 4
adapted case
data copied from adaptation case
data copied from 'best-matching' case
matching specifications
still unsatisfied specification
empty data slot
specified data or similar to specified data
data from 'best-matching' case
data from adaptation case
data similar to 'best-matching' case

Figure 4.6. Case adaptation strategy.

Four steps can be distinguished:

1. A 'best-matching' case is selected based on the similarity of the specifications (mainly of Function data type) as defined in the initial target set.

2. The specifications which are still unsatisfied by the ’best-matching’ case are copied to the adaptation target. Also data are copied from the ’best-matching’ case which are strongly related to the matching specifications. The result of this procedure is that from the adaptation cases which match the still unsatisfied specifications, the cases with Structure and Behaviour data similar to the best-matching case are preferred. So the adaptation case does not only contain a similar problem description (Function), but also a similar solution description (Structure and Behaviour). The idea of this strategy is to keep the differences between the best-matching case and the adaptation case as small as possible.
3. An adaptation case is selected which shows the best similarity with the adaptation target.
4. The adapted case is constructed from data of the best-matching case and the adaptation case. Most of the data comes from the best-matching case. Only the Structure and Behaviour data which is strongly related to the matching specifications of the adaptation case, is copied to the adapted case.

A major issue is which Structure and Behaviour data to copy from the best-matching case to the adaptation target set; as mentioned in step 2. These data are strongly related to the matching Function parameters of the initial target set.

The idea behind this procedure is that the Function parameters represent the problem description, and that the Structure and Behaviour parameters describe the solution to the problem. A strong relation between parameters of these classes indicate a relation between a sub-problem and a sub-solution. So when a case partly matches the specifications in the target set, the Structure and Behaviour parameter that are strongly related to the matching Function parameters will probably form a sub-solution to the (design) problem. It is sensible to use this sub-solution as part of the adapted case, i.e. the output of the CBR-module. Therefore it is also sensible to search for adaptation cases which have as much in common with the sub-solutions as possible. This will reduce the inconsistencies between the initial best-matching case and the adaptation case as much as possible.

It is not evident to reveal these strong relations between the Structure, Behaviour and Function parameters, however. They represent rules of thumb and are very general, so have to be viewed with caution. Practically all parameters are related to each other, and the designer should try to pick the strongest relationships.

Another issue is the inconsistency of the data of the adapted case, especially of the data which do not seem to be important at this moment. For instance some Structure data are required to model the aircraft concept in the Geometrical module, but are not interesting to be reasoned about in this design phase. For example the longitudinal location of the wing. Most of these data are copied from the ’best-matching’ case without giving attention.

Some of these Structure parameters may have to be changed due to the proposed adaptation, however, to remain being realistic. For example, the longitudinal location of the wing will be more forward when engines are mounted on the wing, than when the engines are mounted on the rear fuselage; this is caused by the change of location of the centre of gravity. So, when configuration parameters are copied from the adaptation case, other related Structure parameters have to be copied also.

4.4 Functional module

4.4.1 Tasks

The CBR module has generated an initial aircraft concept. This concept may contain inconsistent data, when it has been adapted, and will probably satisfy only part of the specifications. The Functional module provides the means to analyse, evaluate and modify the initial concept, in order to design a feasible aircraft concept.

With aid of the Functional module the user generates a numerical link between parameters, for example between the take-off distance (functional parameter) and the wing area (primary structural parameter). With the link parameter studies can be performed in order to choose feasible values for the primary parameters.

In contrast to existing numerical tools, such as ADAS [Bil, 1988], the numerical link does not act as a black box. The designer constructs the link himself, with support of the Functional module. This module uses rule-based reasoning to suggest the appropriate algebraic rules and to combine them into a numerical network. The algebraic rules have been put in a knowledge-base in advance. They express the heuristics and simplified physics such as collected in [Roskam, 1990] and [Torenbeek, 1982]; an example is the well-known Bréquet range equation.

Figure 4.7 shows a very simple example of such a numerical network.

Dependent on the completeness of the knowledge-base, the module can be used for a wide variety of design problems. The designer selects which parameters are to be calculated, and in close cooperation the module creates the numerical network. This makes the module very flexible.

4.4.2 QUAESTOR: the implementation

The module is implemented in QUAESTOR [Van Hees, 1997], an expert governed system for the assembly, execution and maintenance of parametric design models of ships. This tool has been developed to support ship designers in the early phases of design by improving access to and control over design related knowledge.

Figure 4.7 shows an example of a very simple numerical link. Figure 4.7 is called a `template` in QUAESTOR-terminology. The `template` is constructed from three types of elements:

- `relations`

  A `relation` is a numerical expression in algebraic notation. That notation has to be specified in the form: `parameter`=*function*(`parameters`); the left clause should contain only one `parameter`, the right clause should contain a function with zero or more `parameters`.

  The `relations` implement the knowledge which has been made explicit, in the form of definition formulae and empirical functions. Because empirical functions are based on statistics, they have a limited validity. This is incorporated by `constraints`.
- `constraints`

  A `constraint` is coupled to one or more `relations`. Only when the `constraint` is found to be `True`, the coupled `relation` will be considered as an appropriate knowledge element.

  A `constraint` is an expression which can be `True` or `False`. The syntax of that expression is less strict than that of a `relation`; it may include inequalities and logic operators.
- `parameters`

  Every `relation` and `constraint` contains `parameters`. The `relations` are selected because of the associated `parameters`; `relations` link `parameters` to each other.

  A `parameters`-value can be a number or a string. `Parameters` in the `template` are either dependent or independent, i.e. they are calculated by `relations` or they are input, respectively. Dependent `parameters` can be divided into top-goal `parameters` and sub-goal `parameters`. The top-goal `parameters` form the target of the link: they are the reason why the link is constructed. The sub-goal `parameters` have been introduced during the construction of the link and are necessary to calculate the top-goal `parameters`.

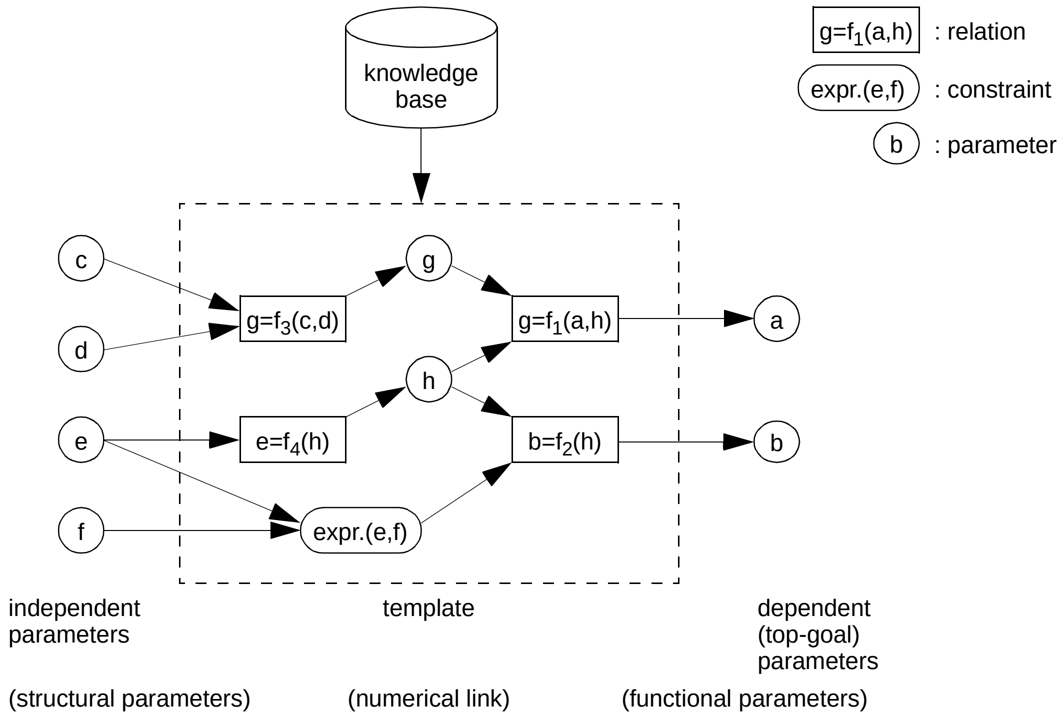


Figure 4.7. A QUAESTOR `template`*, linking independent with dependent* `parameters`*.*

QUAESTOR uses rule-based reasoning techniques to construct the template. The design engineer starts a QUAESTOR session by indicating which knowledge base to use and which parameters have to be calculated, i.e. the top-goal parameters. With backward-reasoning QUAESTOR suggests the relations which can be used to calculate the goal parameter. That is, QUAESTOR searches the knowledge base for relations which contain the goal parameter. A valuable characteristic of QUAESTOR is that also those relations are suggested with the goal parameter in the right clause of the algebraic notation. The design engineer chooses one of the suggested relations.

QUAESTOR also offers the possibility to use external applications. The external application is considered as an external procedure, from which the input and output parameters are fixed. Such an applications is considered to be a type of `relation`. This makes it possible to use existing, reliable applications as part of the template.

The selection of a `relation` often introduces new `parameters`. When these `parameters` are assigned to be dependent, they become sub-goals for which QUAESTOR attempts to find appropriate `relations`. The design engineer can also make them independent by directly giving them values, and QUAESTOR will focus on the remaining new `parameters`. The construction of the `template` has been finished when every `parameter` can be calculated or has been given a value, that is when the number of selected `relations` equals the number of dependent (top-goal and sub-goal) `parameters`.

Already during the construction of the `template` QUAESTOR attempts to calculate the `parameters`' values. Only when the construction has been completed, QUAESTOR is able to solve the entire `template`. The set of `relations` is solved by using the Newton Raphson and the simplex methods. With these methods it is also possible to deal with loops in the `template`.

The designer can easily vary the values of the `parameters` by changing the input. This makes QUAESTOR very suitable to perform parameter studies. The results of such a study can be displayed in a diagram. QUAESTOR is able to display a diagram for one `template` only. When the designer wants to put the results of several `templates` in one diagram, he has to use other tools, for example the Central User Interface. Therefore the QUAESTOR results can be written to external ascii-files. These external files can also be used for other purposes, such as input data for the Geometrical module.

It is possible to change dependent `parameters` into independent `parameters`, by directly giving them a value. The other way around is also possible by removing the input value. In these cases QUAESTOR will interactively modify the `template` to face the new situation.

4.4.3 Working with QUAESTOR

An important issue within QUAESTOR is the suggestion of the appropriate `relations`; this is called the design strategy (or template construction strategy or control knowledge). In general, QUAESTOR searches in the knowledge-base for `relations` which contain the current goal `parameter`. This general strategy is influenced by local strategic knowledge. This local knowledge is coupled to the knowledge elements in different ways:

- Each `relation` within QUAESTOR can be coupled to a `constraint`. Only when the `constraint` is satisfied, the coupled `relation` can be suggested.
- Each `relation` within QUAESTOR has so-called control-attributes. With these attributes the knowledge engineer can express how the `relation` should be used. For example, that the `relation` should be used one-way only (to calculate the `parameter` in the left clause), or that the `relation` is a physical relation instead of an empirical relation (lower priority).
- Also each `parameter` has control-attributes to give information about its use. These attributes for example express the range of values the `parameter` may have, whether or not it should always be calculated or given by the design engineer, etc..

So, within QUAESTOR three modes of operation can be distinguished:

- construction of the knowledge base
  The knowledge base is built by knowledge engineers before a design session is started. The knowledge engineer should be an expert in his domain, because the control attributes of the knowledge elements affect the inference engine of QUAESTOR. Several experts can build one knowledge base.
- construction of the `template`
  A design engineer starts the construction of a `template` by selecting a knowledge base and indicating one or more top-goal `parameters`. The design engineer selects `relations` which are suggested by QUAESTOR, and QUAESTOR builds the `template`. The design engineer should not change the contents of the knowledge base, only use it. He does not need to be as experienced as the knowledge engineer, but familiarity with the knowledge base is highly recommended to construct the `template` efficiently.
- solving the `template`
  While constructing the `template`, QUAESTOR attempts to solve the `template`. This will succeed when the number of dependent `parameters` equals the number of `relations` in the `template`.
  When a `parameter` is calculated, QUAESTOR evaluates if other `parameters` in the `template` can also be calculated. When this is the case, QUAESTOR continues this forward-reasoning process with the newly calculated `parameter`.

4.4.4 Implementation aspects of QUAESTOR

As has been made clear in the previous sections, QUAESTOR is only able to deal with domain knowledge which is available in the form of explicit numerical relations. Much of this knowledge has been formulated, collected and evaluated in [Torenbeek, 1984] and the *Airplane design* books of Roskam. When implementing this explicit domain knowledge in QUAESTOR, some practical issues arise.

Overview of the template

The current version of QUAESTOR has a limited graphical interface. Schemes of the `template`, such as Figure 4.7, are not created. These schemes would also become very complicated when the `template` is built of hundreds of `relations`. This makes it hard for the designer to keep track of the `template` construction process, despite some textual-based support-facilities to present the (incomplete) `template`.

Therefore, the design engineer should be familiar with the knowledge base when constructing the `template`.

Selecting the top-goal parameters

When the knowledge-base has been loaded, the design engineer has to select the top-goal `parameters`. QUAESTOR does not support that task; the designer should know which parameters are important to be subjected to parameter-studies.

This is not a problem when a `template` is needed to transform any set of geometrical `parameters` to the definition parameters of the Geometrical module. In that case the definition parameters are the top-goal `parameters`.

Performing parameter studies

When the `template` is completed, parameter studies can be performed. The designer can easily give multiple input-values for several `parameters`. For each combination the `template` is calculated, and the results are presented in tables.

When problems arise due to the changing values of `parameters`, for example when a `constraint` is not satisfied, QUAESTOR notifies the designer but continues without changing the `template`.

Showing the results

The results of the parameter studies are numbers, rearranged in tables. They can be saved in external files in a specified format called Telitab [Van Hees, 1997]. QUAESTOR also offers the possibility to present the results in a graph. However, only the results of one QUAESTOR session can be presented, i.e. the results accomplished with one `template`. When the results of different `templates` need to be compared, they have to be saved in external files and graphically presented by the Central User Interface.

4.5 Geometrical module

4.5.1 Tasks

The main task of the Geometrical module is to generate a solid model of the concept, in order to check for geometrical inconsistencies and to visualise the concept. The visualisation improves the understanding of the (geometrical) numbers coming from the case and/or the Functional module.

The module can output the model in a number of formats: traditional two-dimensional side-views, a rendered three-dimensional picture, and several graphical formats. These graphical formats can be used by other, more detailed computer programs for more complex calculations.

When the concept is modelled, some typical geometrical parameters can be deduced, such as surface areas and centres of gravity. These parameters can be used by the Functional module or in next design phases.

The model is generated with a minimum of user intervention, so that the designer does not have to be an expert in the applied software. To automate the modelling process, use is made of pre-defined standard components, such as the fuselage, wing, engines, etc.. Also the way these components are located with respect to each other, is pre-defined.

It is possible to pre-define these geometrical constraints, because all possible configurations are known in advance. This is because the configurations are generated by the cases, which have already been stored in the case-base.

4.5.2 Implementation in Pro/ENGINEER

The Geometrical module is implemented in Pro/ENGINEER, a feature-based solid modeller [PTC, 1999]. With this commercial package solids are constructed from engineering features, such as holes, protrusions, rounds, etc.. The way these features are defined appeal to the designer’s intentions: for example a hole can be defined to go through a solid, with its axis halfway one side of the solid.

The features are combined into parts, which can be combined into assemblies. The features are defined and located by parameters and by constraints between parameters and features. With these relations Pro/ENGINEER is able to preserve the consistency of the model when parameter values are changed.

Parameter values can be given directly by the user, or can be read from an external file. When parameter values have been changed, the module will update the model automatically. Error warnings will appear when inconsistencies have been detected.

The automatic generation of the aircraft model is feasible by the use of the Pro/ENGINEER features, its assembly-part-feature hierarchy, the internal relations, and by employing proper definition parameters.

In the Geometrical module the aircraft concept is modelled as a super-assembly which consists of sub-assemblies. They represent the standard components such as the fuselage and wing; see Figure 4.8. A sub-assembly consist of one or more parts. In each part a standard component is defined differently, i.e. each part represents a different component configuration. For example, a turbofan engine is defined differently from a turboprop. A part is defined by features and relations between the features.

It is not possible to use the same part definition more than once with different sets of parameter values. In that case the part definition has to be copied; see for example the Straight tapered Wing, Hor_tail and Vert_tail in Figure 4.8. These parts are defined by the same features.

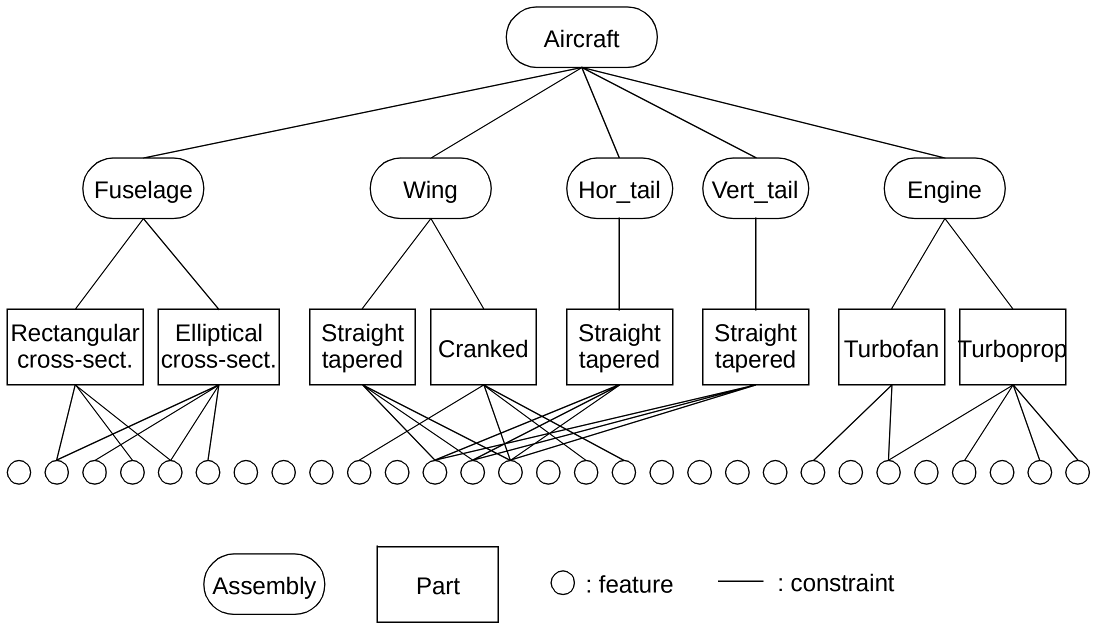


Figure 4.8. The hierarchical structure of the aircraft concept model.

Each feature requires a specific set of parameters. In an internal programming language of Pro/ENGINEER explicit algebraic relations can be defined between parameters. These continuous parameters can be altered as long as the relations are not violated. The relations are defined at every level: super-assembly, sub-assembly and part.

Changing the configuration of a component is a discrete operation on the sub-assembly level. In the Geometrical module this is implemented by choosing a part within a sub-assembly. Additional symbolic parameters are defined in the internal programming language which represent a configuration choice, i.e. they indicate which pre-defined parts to use. Logical expressions are used to suppress the undesired parts. For example, when the type of engine parameter is set to Turboprop, the part Turbofan is suppressed in the model.

Other discrete operations are the changing of the component locations with respect to each other. These are operations on the super-assembly level. When the engines are mounted on the wing, for example, the designer is interested in other location parameters than when they are mounted on the fuselage. For efficiency the previous ’suppress’-method is not applied to model different location definitions. That is, within the aircraft model one set of parameters is used to define the component locations. However, the designer can apply different sets of definition parameters which are internally be transformed into the proper definition set, using the internal programming language.

The set of parameters which define the aircraft model have been selected carefully. Many non-dimensional parameters have been used, because they are less dependent on the aircraft size and appeal more to its shape. When dimensional parameters have to be used, those are selected which have some meaning to the designer, such as the wing area. The definition parameters are also used in the cases in the Case-based reasoning module.

4.6 Central User Interface module

4.6.1 Tasks

Within the AIDA set-up the Central User Interface module should ensure all interactions between the modules and the user. However, the development of AIDA has not been finished yet. The development has reached the stage in which its basic principles can be evaluated; which is the purpose of this thesis project.

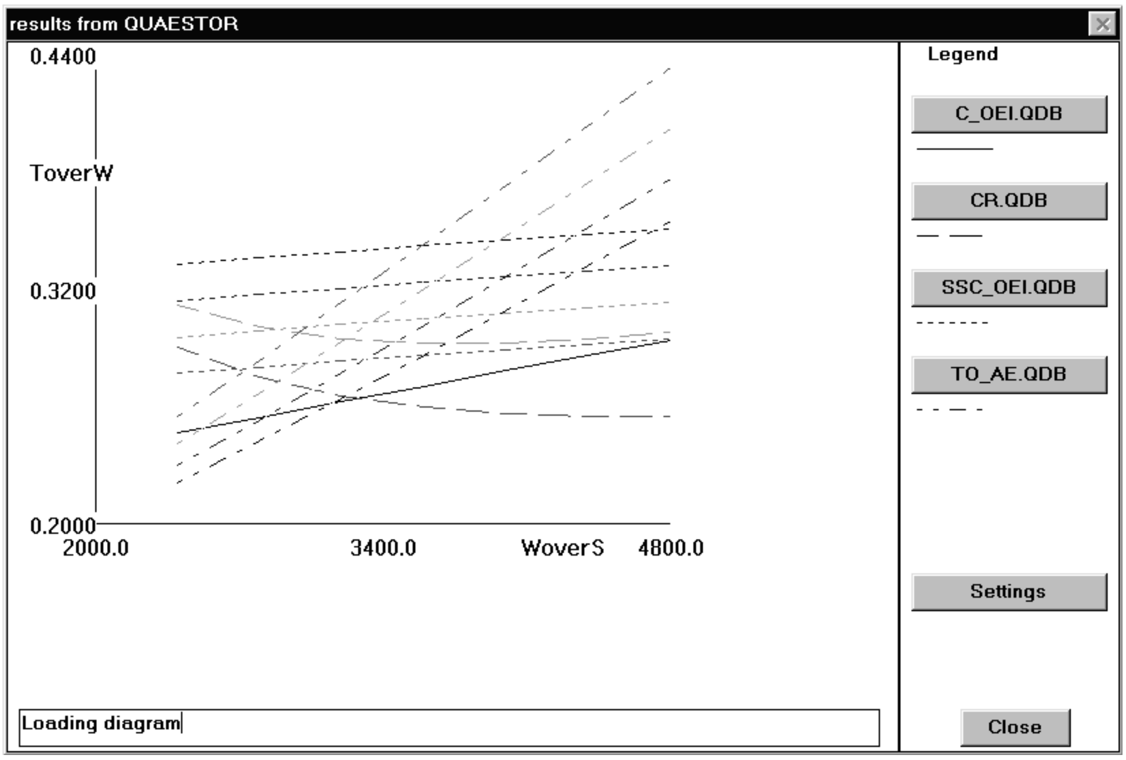
The main function of the CUI is to make the AIDA tool more user-friendly. Because this is not the main concern in this development phase of AIDA, little effort has been put in the development of the CUI. Only some simple tasks have been considered which necessity have directly been unveiled by the other three modules.

One problem is the format of the data-files generated by the CBR module and required by the Geometrical module. Both modules are developed on Unix-platforms, so direct communication should be possible. The CUI transforms the file with the adapted case-data to a file which is readable by the Geometrical module.

Another problem is the limited capability of the Functional module to show the results of parameter studies in diagrams. The CUI can read a number of output-files of the Functional module and combine them into one diagram.

4.6.2 Implementation in C-language

The data-transformation task and the diagram task are implemented in the C-language. With buttons the user is guided to the appropriate menus. Figure 4.9 shows an example of a diagram with the results of four parameter studies. The implementation was carried out by Schipperijn as part of his Master's project [Schipperijn, 1998].



Loading diagram

Figure 4.9. Example of a diagram showing the results saved in four data-files.

5 Design of an aircraft concept with AIDA

5.1 Introduction

To clarify the working of the AIDA system, a design session is described and the supporting role of the AIDA-modules in that process. Each design step is discussed separately, involving one AIDA-module per step.

The case study is based on a design study which has been carried out before in a Garteur-project [Fransen, 1996]. The aircraft specifications are described in Section 5.2.

Based on the specifications an existing aircraft is selected, which will be the starting point for the next design steps. The CBR-module supports the designer with the selection. This is discussed in Section 5.3.

In Section 5.4 the initial case is adapted by integrating parts of other cases. Also in this design step the CBR-module is utilized. The section describes the complex process of selection and integration of the cases.

Next the Functional module is applied to evaluate the adapted case and add some parameters. Section 5.5 describes how the numerical network is generated which is used for this.

The application of the Geometrical module is described in Section 5.6. This module generates a solid model of the designed aircraft concept.

Finally, in Section 5.7 the overall procedure is evaluated, including the way the modules are used and their general characteristics.

5.2 Design specifications

To test the functionality of the AIDA-system the design specifications and data of an earlier study on multi-disciplinary wing optimization [Fransen, 1996] have been used. That study concerned a regional transport airliner intended to carry 70 passengers over a distance of 2500 km, with a cruise Mach-number of 0.70 at an altitude of 30,000 ft. A description of the design specifications for this aircraft is given in Table 5.1. The new design has to meet this list of requirements.

Some specifications of [Fransen, 1996] will be left out in order to clarify the tasks of the relevant AIDA modules. For example the propulsion will not be specified as well as the number of aisles and passengers abreast of the fuselage lay-out. The CBR-module will be applied to select these parameters.

Table 5.1. The (numerical) design specifications of [Fransen, 1996], used for this case-study.

Range & capacity	- 70 pass. plus luggage and 1400 kg freight:	N_pax= 70
	→ max. payload = 8400 kg:	W_pay= 8400 kg
	- range with max. payload = 2500 km:	R_maxPay= 2500 km
Propulsion	- twin-engine:	N_eng= 2
	- take-off thrust = ca. 60 kN:	Tmax= 60 kN
	- high by-pass turbofans	
Fuselage lay-out	- pressure cabin altitude of 8000 ft (2440 m)	
	- max. cruise altitude = 35.000 ft (10670 m):	alt_max= 10,670 m
	- seat pitch = 32 inch (all-economy):	pitch = 32"
	- aisle(s) width at least 20 inch:	aisle_width= 20"
	- 5 passengers abreast:	pass_abreast= 5
	- one aisle:	nr_of_aisles= 1

Performance	- max. cruise Mach-number = 0.70 at 30,000 ft:	Mcruise= 0.70
		alt_cruise= 9150 m

	- max. take-off fieldlength = 1400 m (SL, ISA) at max. take-off weight:	to_length= 1400 m
	- max. landing fieldlength = 1200 m (SL, ISA) at max. landing weight:	l_length= 1200 m
	- max. landing weight = 95% of max. take-off weight:	MLW_ratio=0.95
	- min. ceiling at max. take-off weight after engine failure = 15,000 ft:	alt_OEI= 4570 m

This set of specifications has been chosen because comparable aircraft show a wide variety of configurations (different types of engines, different wing and tail configurations), which seems to make the decisions about the configuration not straightforward. Another reason is that some data which are hard to find have already been collected. Also the development of a 70 passenger airliner seems to be the focus of several manufacturers today.

5.3 Initial estimations with the CBR-module

5.3.1 Introduction

As mentioned in Section 4.3, no explicit knowledge is available to deduce an aircraft concept straight from the design specifications. Therefore the CBR-module of the

AIDA system is used to select a good candidate from existing aircraft that can serve as a starting point for the rest of the design process.

The CBR-module ranks the cases according to their similarity with the specifications, and the designer selects one of them. This is usually the best-matching case, or a case which offers the best flexibility to be modified and adapted in later design phases.

To be able to apply the CBR-module, some work has to be done in advance. This pre-processing work concerns the creation of the classification network and the case-base with data of existing aircraft, as has been described in Section 4.3. The next section in this chapter, Section 5.3.2, discusses this pre-processing work.

When this pre-processing has been finished, the designer puts the design specifications into the target-file. The CBR-module compares each case in the case-base with the data in the target- file, and expresses their similarity by a number. Section 5.3.3 describes how the target-file is deduced from the design specifications of Section 5.2.

The output of the CBR-module is described in Section 5.3.4. It shows the results of the similarity measurements. In the next section, Section 5.3.5, it is discussed how to deal with the output; i.e. which existing aircraft concept to select.

5.3.2 The pre-processing work

In the pre-processing work an index network is created which links each case and its data with the parameter domains; this process has already been described in Section 4.3. This work is done in advance in order to save time when the CBR-module is actually used to determine the similarity between the target-file and the cases.

Three aspects are involved in this process:

- defining the hierarchical classification structure of the data
- creating case-files with the appropriate case-data
- defining the parameter domains

Classification structure

The classification structure is based on the Function-Behaviour-Structure view of Rosenman and Gero [Rosenman, 1994]. This has been described in Section 3.2.7.

The Function is associated with the performances of the aircraft, such as: range, payload, speed, take-off length, ceiling, etc.. These parameters have been grouped together into logical combinations, like the payload and the range. In Figure 5.1 the hierarchical classification structure of the Function class has been sketched.

The Structure is associated with the physical description of the aircraft concept. The aircraft is built of major components, such as the fuselage, wing, powerplant, etc.. The attributes of the classes define the geometry of these components and their

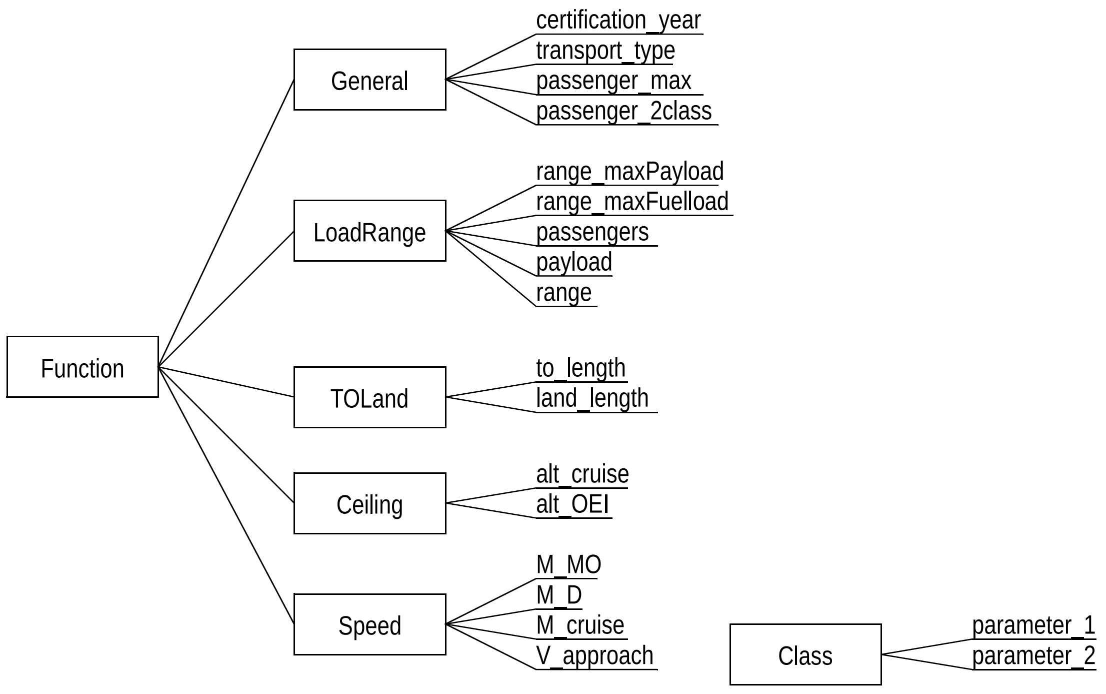


Figure 5.1. The Function classification structure.

location with respect to other major components. When extra attributes are necessary to define a specialization of a component, a sub-class is created. For example, the class engine contains the sub-classes turbofan and turboprop. In a separate class the lay-out of the overall aircraft are defined, such as the wing configuration, number of engines, etc.. Figure 5.2 shows a part of the classification structure of the Structure class.

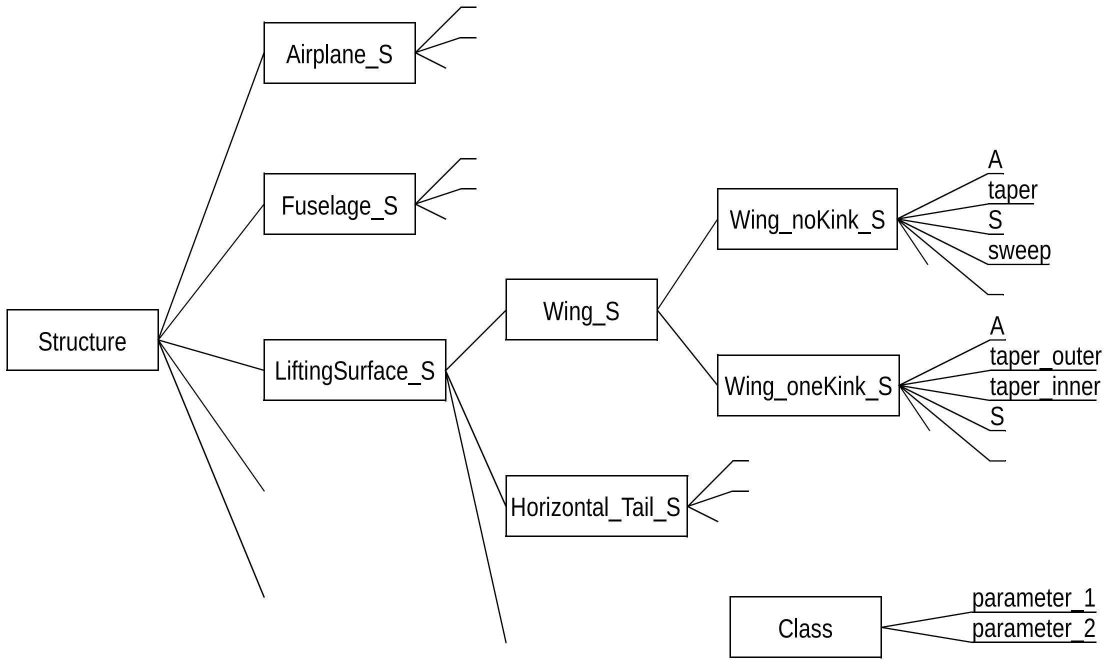


Figure 5.2. Part of the Structure classification structure.

The Behaviour class is the class between the Function and the Structure. Behaviour parameters are associated with the parameters which describe certain qualities of the aircraft and its components, such as the aerodynamics and weight. Attributes which clearly belong to a component, are grouped together into separate classes. This is sketched in Figure 5.3, which shows the Behaviour classification structure.

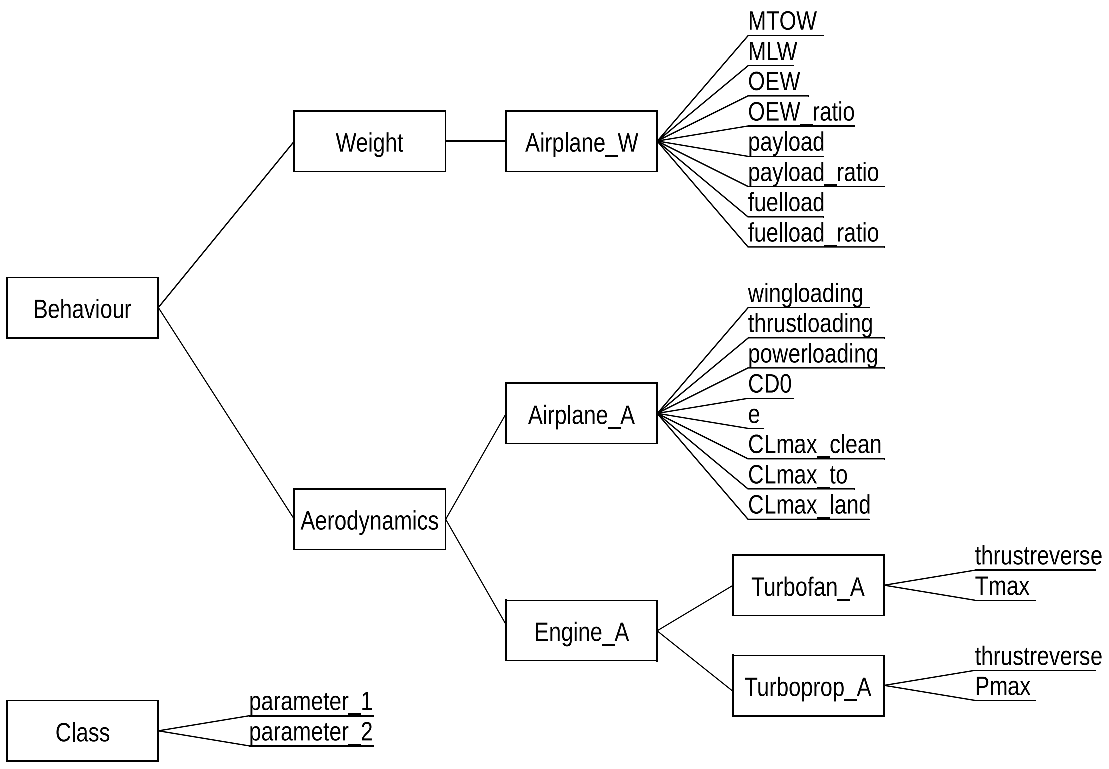


Figure 5.3. The Behaviour classification structure.

The classification structure is defined in a separate class-file.

Case-files

The data in the cases are organized according to the described classification structure. The data is copied from sources such as [Jane's, yearly]. It is noted that only data is collected of existing aircraft, not of ongoing projects. This is to ensure that the case-base contains data which have proven to be realistic.

To be able to compare the data, attention has to be paid to the parameter definitions. These definitions may differ from case to case, because different sources have been used. Therefore the Structure data is checked with the available 3-view drawings of the aircraft. Especially the surface areas are checked, and are determined according to their definitions in [Torenbeek, 1982]. The Structure data which is not explicitly mentioned in the sources, has been derived from drawings; such as the location of the wing and tail surfaces.

Some parameter values strongly depend on the aircraft operating conditions. The maximum liftcoefficient, for example, will be different for clean wings and for wings with flaps in take-off position. When the conditions can unambiguously be pre-defined, separate parameters are used, such as CLmax_clean and CLmax_to; see the subclass Behaviour.Aerodynamics.Airplane_A in Figure 5.3.

For some parameters, however, the conditions are hard to pre-define. In those situations the conditions are described by other parameters. For example the range is often given in combination with a payload which is not the maximum, i.e. not a pre-defined condition. In that situation both the range and the load have to be considered when similarities between the aircraft and the specifications are examined. The CBR-module is capable of doing that when both parameters are organized into one class; see the subclass Function.LoadRange in Figure 5.1.

However, it is tried to work with pre-defined conditions as much as possible, because this will simplify the similarity measurements.

Figure 5.4 shows a part of a case-file of the Boeing 777-200. The data shows that two range-payload combinations have been found: range= 6685 km with 375 passengers, and range= 10723 km with 305 passengers. These combinations are organized in separate objects of the same class.

The numbers 1, 2 and 5 in the second column indicate the type of parameters. Also the source of information is captured by the source-parameter.

```
    .
    .
    .
Function
OBJECT     B777_200RangeCombi1
CLASS      LoadRange
PARENT     B777__200
           source     2          zesA
           pass       5          375
           range      1          6685

Function
OBJECT     B777_200RangeCombi2
CLASS      LoadRange
PARENT     B777__200
           source     2          zesB
           pass       5          305
           range      1          10723

Function
OBJECT     B777_200ceiling
CLASS      Ceiling
PARENT     B777_200
           source     2          zes
           alt_cruise 1          10000.000
           alt_OEI    1          6000.000
    .
    .
    .
```

Figure 5.4.   Part of the case-file of the Boeing 777-200.

The parameters alt_cruise and alt_OEI in Figure 5.4 suggest high accuracies; this is not true, however. The three zeroes have been added to mark the parameters that their values have been estimated by the case-base developer, and do not come from the regular sources.

The estimated values have been added to complete the case-file. This is important because of the impact of missing data on the matching process of the CBR-module. A case-file can not show a matching of a parameter when that parameter is not included in the file, which therefore may result in a lower similarity with the target-file, i.e. the design specifications.

The estimations of the missing values is based on the knowledge of experts, and with the parameter domains at hand to get an idea of the required accuracy of the estimation. The latter is important to know in case the developer wants to change the parameter domains.

Parameter domains

The matching of the parameter values is not based on their exact values, but on the value domains to which they fit in. This method allows a kind of qualitative matching of the data, which has been described in the previous chapter, Section 4.3. Because the similarity measurements depends on the parameter domains, it is important to define them with care.

The domains are defined in a separate file; a part of it is shown in Figure 5.5.

```
         :
{Cabin_S.nr_of_aisles} = {1 2}
{Cabin_S.nr_of_pass_abreast} = {[1,3] [4,*}
         :
{Wing_S.A} = {[0,6> [6,10> [10,*}
{Wing_S.sweep} = {*,-5> [-5,10> [10,30> [30,*}
         :
{Airplane_S.engine_type} = {turbofan turboprop piston}
{Airplane_S.nr_of_engines} = {1 2 3 4}
         :
{Airplane_W.MTOW} = {[0,6> [6,30> [30,100> [100,350> [350,*}
{Airplane_W.maxPayload} = {[0,10> [10,20> [20,40> [40,*}
         :
{General.pass_max} = {[0,20> [20,90> [90,130> [130,180> [180,300> [300,*}
         :
{LoadRange.pass} = {[0,80> [80,110> [110,150> [150,250> [250,*}
{LoadRange.payload} = {[0,10> [10,20> [20,40> [40,*}
{LoadRange.range} = {[0,1000> [1000,2500> [2500,4500> [4500,7000> [7000,*}
         :
{TOLand.to_length} = {[0,1000> [1000,1500> [1500,2100> [2100,3000> [3000,*}
{TOLand.l_length} = {[0,800> [800,1300> [1300,1800> [1800,2300> [2300,*}
         :
{Ceiling.alt_cruise} = {[0,2400> [2400,11000> [11000,*}
{Ceiling.alt_OEI} = {[0,4000> [4000,*}
         :
{Speed.M_cruise} = {[0,0.6> [0.6,0.8> [0.8,1.0> [1.0,*}
         :
```

Figure 5.5. Part of the domain-file.

The definition of the domains is not always evident, especially for continuous parameters. Arguments based on physics or airworthiness regulations are often not available. In those situations the domains are defined based on their discriminating role: when do you consider two parameter values to be of the same qualification (large, small, etc.), and when not. Too many domains will probably result in very little matchings, and a few large domains will probably not discriminate enough.

The next examples reveal some of the typical problems (see Figure 5.5):

- domains of parameter nr_of_aisles of the subclass Structure.Cabin_S: 1 and 2
  This parameter typically defines whether an aircraft is a wide-body (2 aisles) or not (1 aisle).
- domains of parameter nr_of_pass_abreast of the subclass Structure.Cabin_S: [1,3] [4,∞)
  Cabins with 1 to 3 passengers abreast are very small and should be considered different from cabins with 4 or more passengers abreast. One could argue about the boundary: 4 passenger abreast is also small. So the domains are not so evident.
- domains of parameter A of the subclass Structure.Wing_S: [0,6), [6,10) and [10,∞)
  The aspect ratio A of a wing is considered small when smaller than 6, large when larger than 10, and medium in between these values.

  The aspect ratio directly affects the structural loads on the wing root and induced drag, and therefore has influence on other design parameters such as the wing loading, the root thickness ratio, the range, the Take-off Field Length (to_length), etc. (Chapter 7 of [Torenbeek, 1982]). This wide range of effects does not result in clear parameter domains. The domain boundaries are rather arbitrary and are based on the experience of the developer and the examination of existing aircraft.
- domains of parameter sweep of the subclass Structure.Wing_S: (–∞,-5), [-5,10), [10,30) and [30,∞)
  The sweep angle of the wing has a large influence on the maximum speed of high-subsonic aircraft, and therefore affects the maximum allowable wing thickness. With this in mind the domains define a forward swept wing (–∞,-5), a more or less straight wing [-5,10), a slightly swept wing [10,30), and a highly swept wing [30,∞) for high-subsonic velocities. These boundary values are debatable.
- domains of parameter engine_type of the subclass Structure.Airplane_S: turbofan, turboprop and piston
  This parameter defines the type of engine. It is a discrete parameter with each value clearly representing another type, so each value represent another parameter domain.

- domains of parameter MTOW of the subclass Behaviour.Weight.Airplane_W: [0,5.670), [6,30), [30,100), [100,350) and [350,∞)
  The first domain represents aircraft from the light airplane category according to airworthiness regulations (FAR 23, JAR 23). Therefore these aircraft have to face other design requirements than the heavier aircraft. So this domain is clearly defined. The other domains are again disputable.
- domains of parameter pass_max of the subclass Function.General: [0,20), [20,90), [90,130), [130,180), [180,300) and [300,∞)
  According to airworthiness regulations only 1 cabin person is required when carrying up to and including 19 passengers. This results in a clearly defined, first domain. The other domains represent small to very large (commercial) aircraft, with arguable domain boundaries.

5.3.3 The target-file

The target-file contains a set of parameters and their values. The CBR-module determines the similarity between each case and this set by counting the matching parameter domains.

The target-data is organized in the same classification structure as the case-data structure. This structure makes it possible to determine the similarity for each target-object. The weighted average of the similarity scores of all objects directs the total (global) similarity score of a case.

Next figure shows the target-file as used in this study. It represents the design specifications as has been discussed in Section 5.2.

```
OBJECT PassRangeCombi PRIO 100
LoadRange.pass       5         70
LoadRange.payload    1         8.4
LoadRange.range      1         2500

OBJECT M_cruise PRIO 50
Speed.M_cruise       1         0.70

OBJECT TOLanding PRIO 50
TOLand.to_length     1         1400
TOLand.l_length      1         1200

OBJECT Alt PRIO 50
Ceiling.alt_cruise   1         9150
Ceiling.alt_OEI      1         4570
```

Figure 5.6. The target-file.

When comparing the target-file with the specifications as put forward in Table 5.1, some remarks can be made:

- The specifications on the propulsion are kept out of the target file, as well as the specifications on the fuselage lay-out.

  These design specifications have been the result of some preliminary studies of the other specifications. The experience of the designer and the examination of existing aircraft has led to the (preliminary) conclusions about the propulsion and the fuselage lay-out. However, we want to apply the CBR-technique to come to such conclusions without bias of the designer. Therefore these specifications have not been copied into the target-file.
- The performance specifications have been organized in separate target-objects, each containing related parameters.

  The matching scores will be determined per target-object. Therefore the design specifications which the designer wants to be considered in combination with each other, have been put into one object. These objects are instantiations of the classes which have been discussed in the beginning of this section.
- The load-range specifications are given more importance than the other specifications, as is defined by the different priority values (PRIO).

  The priority number defines the weight of the target-object. These numbers affect the global similarity score of the cases, since they are defined as the weighted average of the similarity scores per target-object. The priority numbers can easily be changed during the usage of the CBR-module, when the designer is not content with the (first) results of the module.

5.3.4 Output of the CBR-module

The results of the CBR-module are shown in Figure 5.7. Only those cases are presented which have a global similarity score of 30% or higher.

Case	Global	PassRangeCombi	M_cruise	TOLanding	Alt
Target prio. →		100	50	50	50
Saab 2000	87	67	100	100	100
Fokker 70	77	67	100	50	100
Embraer 145	64	34	100	50	100
BAe RJ85	60	0	100	100	100
BAe RJ70	60	0	100	100	100
B 737-400	54	34	100	0	100
B 737-500	54	34	100	0	100
B 737-300	54	34	100	0	100
Fokker 100	54	34	100	0	100
Canadair RJ200	54	34	100	0	100
BAe RJ100	50	0	100	50	100
MD C17	50	0	100	50	100
Dornier 328-110	44	34	0	100	50
MD 87	40	0	100	0	100
MD 90-30	40	0	100	0	100
ATR 72-200	34	34	0	100	0
A 321	34	34	0	0	100

Figure 5.7. The results of the similarity measurements between the target-file and the cases.

There are some general aspects which have to be considered when examining the results of Figure 5.7:

- The similarity score on the target object ’PassRangeCombi’ does nowhere exceed the value 67%. This means that maximum two of the three target parameters (’pass’, ’payload’ and ’range’) of that object are matching. This is a direct result of the way the case-data has been organized: the case-objects either contain the parameter combination ’pass’-’range’, or the combination ’payload’-’range’, but not all three parameters. The purpose is to eliminate the effects of the redundancy of the parameters ’pass’ and ’payload’ on the similarity scores.

  This redundancy is included in the case-data because the payload is often expressed by the number of passengers, and sometimes by the weight in tons. To be able to search for comparable aircraft, it must be possible to use both parameters. However, a case should not have a higher similarity score because it contains both parameters expressing a comparable property. So, when both

’payload’ and ’pass’ are known in combination with (the same) ’range’, this data is put into two separate case-objects: one containing ’pass’ and ’range’, the other containing ’payload’ and ’range’. An example is included in next Figure.

```
    .
    .
    .
Function
OBJECT    RJ70RangeCombi1
CLASS     LoadRange
PARENT    RJ70
          source    2         three
          pass      5         94
          range     1         2278

Function
OBJECT    RJ70RangeCombi2
CLASS     LoadRange
PARENT    RJ70
          source    2         three
          payload   1         10.07
          range     1         2278
    .
    .
    .
```

Figure 5.8. Part of the case-file of the British Aerospace Regional Jet 70 (BAe RJ70).

As a result, the priority factor of the object PassRangeCombi will effectively be reduced from 100% to 67%.

- The target parameter ’range’ with value 2500 is linked with the domain [2500,4500); the value is on the edge of the domain. So the British Aerospace RJ70 with a range of 2278 km does not match with the target, but the Fokker 70 with a range of 3407 km does.

  This is a major drawback of this particular CBR-module [Netten, 1997]: applying binary matching criteria and the use of strict domains.
- According to the sources the BAe RJ70 has a range of 2278 km with the maximum payload, which is 10.070 tons or 94 passengers. With the current parameter domains both the range and the payload or number of passengers do not match the target, i.e. the specified parameters (2500 km with 8.4 tonnes payload or 70 passengers). Therefore the BAe RJ70 scores 0% on the object PassRangCombi; see Figure 5.7.

  However, it is reasonable to believe that this aircraft can meet the requested range specification when the payload is reduced. And that it can meet the requested payload with a smaller range. So, the BAe RJ70 is likely to be able to score 34% on the payload-range specification, but the results do not show that.

  It is the designer’s experience which can give that insight; the CBR-technique is not able to do so.
- It is also possible to combine the target parameter ’M_cruise’ with the parameters ’alt_cruise’ and ’alt_OEI’ in a new target-object. This will decrease the importance of ’M_cruise’ with respect to ’alt_cruise’ and ’alt_OEI’. In that

situation the ranking of the slower and faster aircraft will rise, such as the Dornier 328-110 (M_cruise=0.57000 [estimated]) and the Airbus A321 (M_cruise=0.80), respectively.

So, other object-structures in the target-file will result in different scores of the total similarity. This reveals the influence of the target-file structure.

- The priority numbers reflect the designer's opinion that the specifications about the payload-range combination are more important than the other specifications. However, the priority numbers 100 and 50 have been chosen arbitrarily.

  Other priority numbers, for example 100, 50, 70 and 20, respectively, increase the influence of the take-off and landing specifications, and decrease the importance of the altitude specifications. This result in a slightly other ranking of the cases: the Dornier 328-110 will rise due to its matching take-off and landing distances, and the Embraer 145 and the Boeing 737 series will drop for the same reasons.

5.3.5 Selecting the best case

A case will be selected which satisfies all design specifications, or which seems to be easily modified in order to meet the design specifications. So the question is: which case offers the best prospect to fulfil the design specifications, after some adaptations of the concept?

The general idea is that the more a case satisfies the requirements, the less modifications will be needed. Therefore only those cases are considered which have a high similarity score.

However, selecting a case is not just a matter of choosing the best-matching case from Figure 5.7. The difference in similarity scores may be very small, and/or the best-matching case may be difficult to adapt to meet all design specifications.

The selection process can be organized into three steps:

- examine the configurations with respect to the global similarity scores
  It should be kept in mind that the focus of this design phase is to select a proper configuration. So first priority is to see whether there are big differences between the configurations of the best-matching cases at all.
- examine the configurations with respect to the local (object) similarity scores
  When it is not clear which configuration to choose, i.e. when the variety of the configurations is big, it is examined how the similarity ranking has been built. The similarity scores are examined at a lower level, per target-object, in order to find relations between configuration aspects and specifications.
- select the best case
  Considering the examinations of the global and the individual local similarity scores, the designer selects a case, in his opinion the 'best case'.

Examination of the configurations with respect to the global similarity scores

For a clear view on the different configurations, simple sketches have been drawn of the best-matching cases. These sketches show some of the configuration aspects:

- turboprop or turbofan engines
- number of engines
- location of the engines (although the sketches are only 2 dimensional, the spanwise location of the engines clearly suggests the traditional wing-mounted and fuselage-mounted configurations)
- vertical location of the wings
- type of tail
- wide-body or single-aisle fuselage (large or small diameter)

The sketches are shown in Figure 5.9.

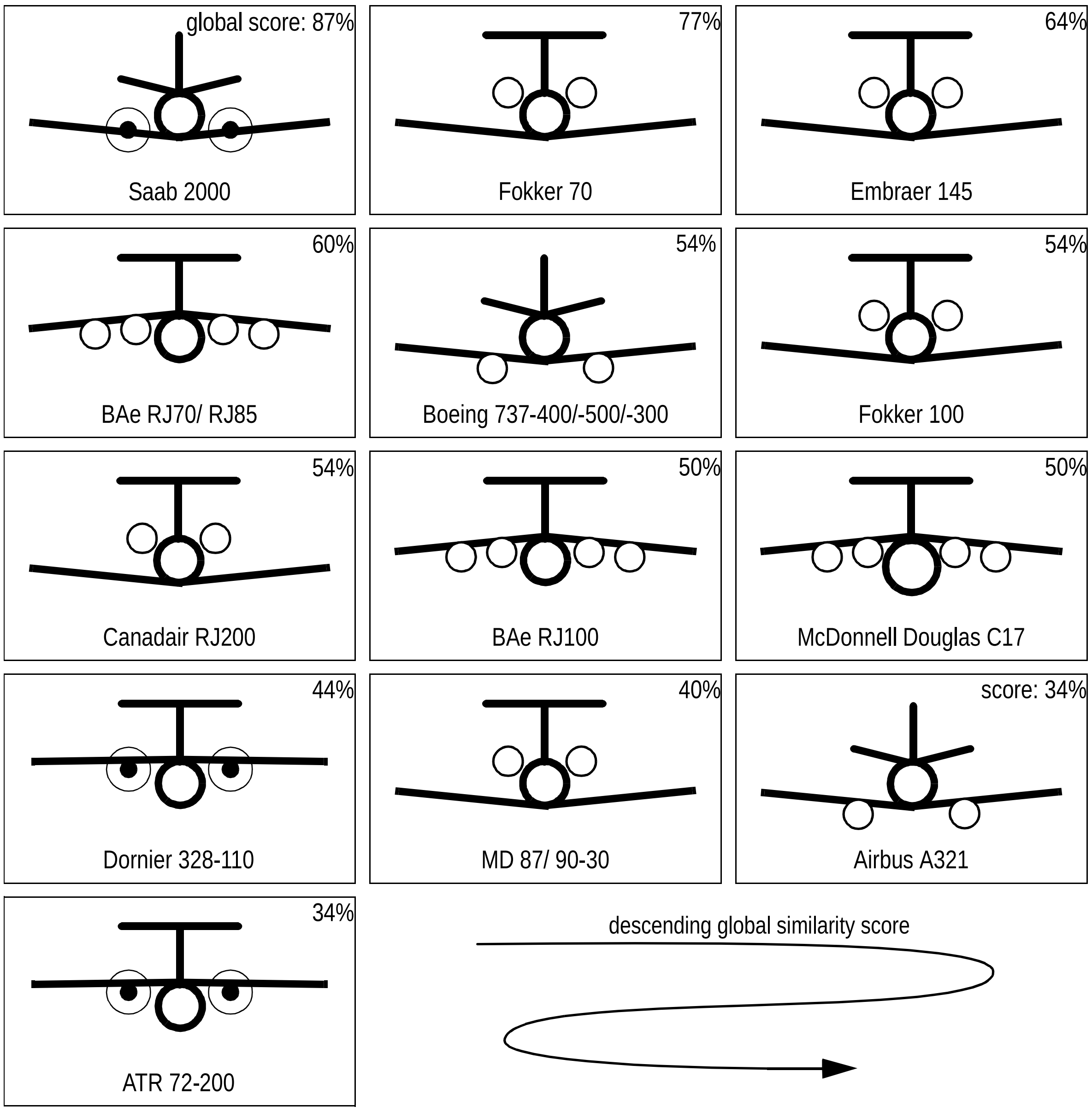


Figure 5.9. The configurations of the best matching cases.

The following can be said about the configurations:

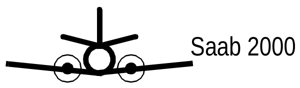
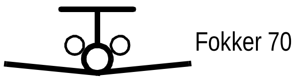
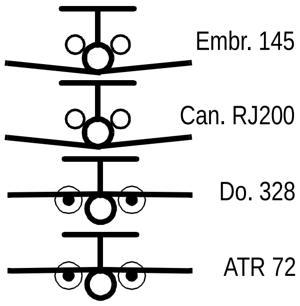
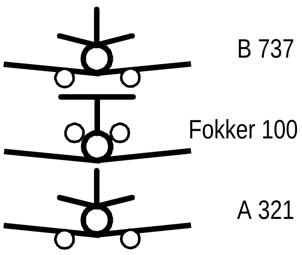
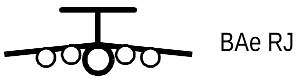
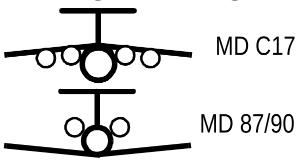
- Five different configurations seem appropriate:
  - 2 fuselage-mounted turbofan engines, T-tail and low wings (6 aircraft: the Fokker 70 and 100, the Embraer 145, the Canadair RJ200, and the MD 90 and 87);
  - 2 wing-mounted turbofan engines, conventional tail and low wings (4 aircraft: the Boeing 737-300, -400, and -500, and the Airbus A321);
  - 4 wing-mounted turbofan engines, T-tail and high wings (4 aircraft: the BAe RJ70, RJ85 and RJ100, and the MD C17);
  - 2 wing-mounted turboprop engines, T-tail and high wings (2 aircraft: the Dornier 328-110 and the ATR 72-200);
  - 2 wing-mounted turboprop engines, conventional tail and low wings (1 aircraft: the Saab 2000);
- The MD C17 is a military cargo aircraft. All other aircraft are commercial passenger airplanes.
- Only the Saab 2000, the Dornier 328 and the ATR 72 have turboprop engines. The other aircraft have turbofans.
- All aircraft have two engines, except for the BAe RJ series and the MD C17; they have four engines.
- Most of the aircraft have their engines mounted on the wing: all turboprop aircraft, all four-engined turbofan aircraft, and the two-engined turbofans of Boeings and Airbus.
- All four-engined aircraft have a high-wing configuration, and so do most of the turboprop aircraft (except for the Saab 2000 turboprop). The other 10 aircraft have low wings.
- All high-wing configuration aircraft have T-tails, and so do the fuselage-mounted engine configuration aircraft. All wing-mounted engine aircraft with low wings have conventional tails.
- Only the MD C17 cargo aircraft has a large-diameter fuselage, comparable with wide-body aircraft. The other aircraft have single-aisle fuselages.

The diversity is big between the configurations of the best-matching cases. Therefore the results of Figure 5.7 will be examined more closely in relation with the configurations.

Examination of the configurations with respect to the local similarity scores

In this sub-section the similarity scores of Figure 5.7 are examined per target-object, with the intention to formulate a preference for a specific case. The configurations are shown with the matching results of the individual target-parameters.

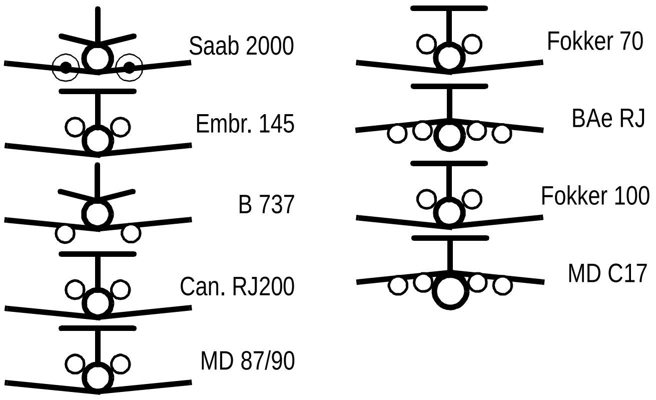
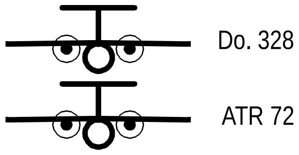
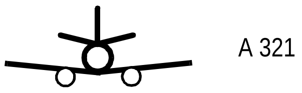
Table 5.2. Similarity scores for the payload/range specifications.

target-object	object score	configurations	
PassRangeCombi	67% (max.)		
	34%	pass/payload: okay range: too short 	pass/payload: too high range: okay 
	0%	pass/payload: too high range: too short 	pass/payload: too high range: too long 

Conclusions with respect to the payload-range specifications:

- The four-engine configuration comes into sight at higher payloads only.
- The same appears for the low-wing-mounted turbofan aircraft: payloads are too high.
- The turboprop aircraft have smaller or similar ranges.
- It is difficult to make general remarks about configuration aspects other than the type of engines.
- There are no aircraft in lower payload domains, because the specified pass/ payload domain is bounded by zero.

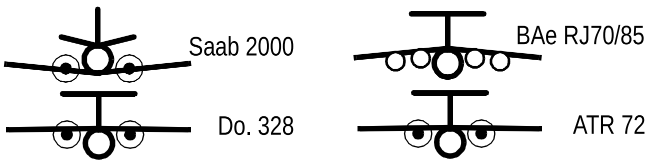
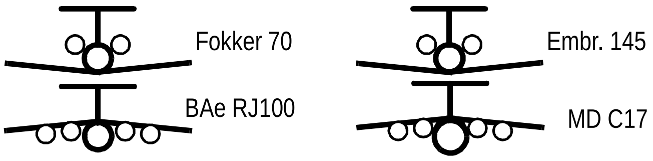
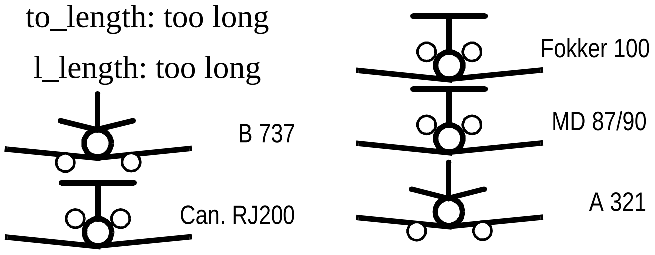
Table 5.3. Similarity scores for the cruising speed specifications.

target-object	object score	configurations	
M_cruise	100%	 Saab 2000 Fokker 70 Embr. 145 BAe RJ B 737 Fokker 100 Can. RJ200 MD C17 MD 87/90	
	0%	M_cruise: too low 	M_cruise: too high 

Only one conclusion is drawn with respect to the cruise speed specifications:

- The turboprop engined aircraft are slower or just meet the specified cruise speed domain.

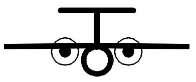
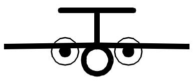
Table 5.4. Similarity scores for the take-off and landing distance specifications.

target-object	object score	configurations
TOLanding	100%	
	50%	to_length: too long l_length: okay 
	0%	to_length: too long l_length: too long 

Conclusions with respect to the take-off and landing distance specifications:

- There are no aircraft in Figure 5.7 with smaller take-off and landing distances (domains).
- The turboprop-powered aircraft all meet the desired take-off and landing performances.
- Most of the high-wing configuration aircraft meet these performances. Only the larger high-wing aircraft need longer take-off distances, such as the BAe RJ100 and the MD C17.
- The configuration with fuselage-mounted engines and the configuration with turbofans mounted on low wings have bigger take-off distances.

Table 5.5. Similarity scores for the altitude specifications.

target-object	object score	configurations
Alt	100%	*all configurations, except:*
	50%	alt_cruise: too low  Do. 328
	0%	alt_cruise: too low alt_OEI: too low  ATR 72

Conclusions with respect to the altitude specifications:

- The specified altitude is reached by all aircraft except for two turboprop powered aircraft. The third turboprop aircraft does reach the required altitude.

Selection of the best case

With the conclusions from the previous sub-section, the Fokker 70 is selected as the ’best-case’. This selection is based on the following arguments:

- turbofan engines seem to be more suitable than turboprop engines
  Although the best-matching aircraft, the Saab 2000, has turboprop engines, the next 10 aircraft with high similarity scores have turbofan engines. When considering the other turboprop aircraft, problems can be expected with the range, the cruising speed and the altitude.
- two engines seem preferable above four engines
  Most high-scoring aircraft have two engines. The four-engine configurations with the highest similarity scores are the BAe RJ70 and RJ85, sharing the 4th ranking with 60% (compared with the other rankings of 87%, 77% and 64%).
  The four-engined configuration is applied to the bigger aircraft; their pass/payload domains are too high.
- from the two-engined turbofan configurations, the Fokker 70 configuration is favourite

The other two-engined turbofan configuration is that of the Boeing and Airbus: the engines mounted on low-wings. These aircraft have lower similarity scores than the Fokker 70 and the Embraer 145. They score especially low on the pass/ payload target and the take-off and landing distance targets.

The fuselage-mounted configurations generally show good similarities with the cruising speed and altitude targets, but score low on the take-off and landing distance targets.

For the sake of completeness, some interesting data of the selected Fokker 70 are copied:

- configuration:
  two turbofan engines mounted on rear-fuselage, low wings, T-tail, single aisle;
- performances:
  - passenger/range combination (matching domains): 79/3407 km;
  - cruise Mach number (matching domain): 0.77 (estimated);
  - take-off and landing distances (take-off distance too long): 1573m and 1274m;
  - cruise and OEI (One Engine Inoperative) altitudes (matching domains): 10,670 m and 6,000 m (estimated);
- weights:
  - MTOW (Maximum Take-Off Weight): 39.92 tons;
  - OEW (Operational Empty Weight): 22.78 tons (ratio with MTOW: 0.57);
  - max. payload: 10.78 tons (ratio with MTOW: 0.27);
- dimensions:
  - fuselage length: 27.88 m;
  - fuselage diameter: 3.30 m;
  - number of passengers abreast: 5;
  - wing span: 28.08 m;
  - wing area: 97.9 m^2;
  - wing taper: 0.21;
  - wing aspect ratio: 8.4;
  - wing sweep: 18 degrees (outer part);
  - wing thickness ratios on root and on tip: 0.123 and 0.096;
  - horizontal tail area: 21.72 m^2;
  - horizontal tail volume coefficient: 0.74;
  - vertical tail area: 15.2 m^2;
  - vertical tail volume coefficient: 0.065.

5.3.6 Remarks

There are some reflections that may have played a role in the selection, but which do not refer to the presented results only. They originate from the designer’s general knowledge about the aircraft and from talkings with professor Torenbeek.

About the Saab 2000

Although the Saab 2000 has the highest similarity score, it is not selected for further elaboration in this case study. Instead the Fokker 70 has been selected as the single case to proceed with. In a real design process several alternatives are studied simultaneously, however. In that situation the Saab 2000 would probably be one of those alternatives.

One pragmatic argument for the selection of the Fokker 70 in this case study is that it reveals some typical characteristics of the applied design procedure, such as an original lay-out as a result of the adaptation phase (Section 5.4). Another argument was the availability of turbofan performance data; see next item.

A critical remark about the Saab 2000 similarity score is the passenger load. With a passenger load of 50 in a 2-class seating configuration and 58 as a maximum, the Saab 2000 matches the parameter domain [0,80) which is related with the specified passenger number of 70. One may argue whether this matching is appropriate: is a 50 passenger aircraft comparable with a 70 passenger aircraft? If not, the domain boundaries need to be revised. This kind of question is always present when the boundaries have been arbitrary chosen.

About the turbofan engines

Originally, the specifications included a preference for turbofan engines; see Section 5.3.3. When considering the quantitative specifications only, however, the CBR-module reveals that turboprop engines may be a good alternative, as part of the Saab 2000 configuration.

Nevertheless, in the presented case study the turbofan engine is selected. One of the reasons is that performance data of a modern turbofan engine was available. When this type of engine appears to be inappropriate, however, it is always possible to select the turboprop engine as well.

About the MD C17

The MD C17 is a military cargo aircraft, which generally has to meet other specifications than commercial aircraft. Some of these design specifications are hard to quantify, but have a large impact on the configuration. For example the high-wing configuration is applied to minimize the distance between the cargo floor and the ground, for easy access of the cargo. However, the inclusion of this military aircraft in the list of best-matching cases may give the designer another view on the design problem.

The high-wing configuration also allows for good take-off and landing performances, which was another important design specification of the MD C17.

That is why its landing distance matches the landing distance target, despite of its much higher weight than the other aircraft.

About the BAe RJ70

As the type-code suggests, the BAe RJ70 is intended to carry 70 passengers in a 2-class seating configuration. Though this is equal to the specifications, the results in Figure 5.7 show no matching of the target-parameter ’pass’.

This is because the range is given for the maximum number of passengers, which is 94 passengers; this is in a single class seating configuration. The number of 94 passengers is outside the required parameter domain of [0,80) (which has arbitrary been defined).

Another item is the range. The BAe RJ70 is able to reach 2278 km with 94 passengers. This range is close to the specified range of 2500 km, but is outside the related parameter domain of [2500, 4500). These domain boundaries have been chosen arbitrary, though, in the pre-processing phase. Other realistic domain borders could have resulted in a matching of the range of the BAe RJ70.

The selection arguments

The arguments which have been used to select the ’best-case’, as described at the end of the previous section, are only focused on the data itself; it is tried to use as little domain knowledge as possible.

Consider for example the argument that the fuselage-mounted engine configuration of the Fokker 70 is in favour of the wing-mounted engine configuration of the Boeing and Airbus aircraft. The preference is purely based on better global similarity scores of the Fokker 70 and the Embraer 145 (see Figure 5.9). The similarity scores per case-object (see Table 5.2 to Table 5.5) are considered to better understand the differences of the global scores. This does *not* imply that physical explanations are tried to be found, however.

Intentionally the domain knowledge is consulted as less as possible, because the large number of physical arguments which may be involved makes process of making design decisions very complicated. This was one of the reasons to apply CBR-techniques: to make this process simpler.

5.4 Adaptations with the CBR-module

5.4.1 Introduction

The selected case, the Fokker 70, does not match all target data. The purpose of this adaptation phase is to improve the configuration of the Fokker 70, if possible, so that it becomes a better starting point for the next design steps.

The focus of the adaptation step is on the configuration of the aircraft. Some other data is involved with have close relationships with the configuration.

The CBR-module has been set up in such way that the pre-processing work does not have to be rehearsed. This means that the same indexing network can be used, with the same case-files, domain file and classification file. Only the target-file has to be defined once more. This is a complex topic, though, and is extensively described in Section 5.4.2.

The other sections describe which adaptations will be performed and for what reasons. This includes the examination of the similarity score for the adaptation target, Section 5.4.3, and the choice of the adaptations, Section 5.4.4.

The other section, Section 5.4.5, describes how the adaptations are actually worked out, and which practical issues are involved.

5.4.2 The adaptation target-file

As has been described in Section 4.3.4 of the previous chapter (step 2), the adaptation target-file consists of two types of data:

- data representing those design requirements which have not been met by the ’best case’

  This is the main part of the target-file. The purpose is to find cases which do meet those specifications, and use parts of these cases to adapt the original ’best case’. The resulting (adapted) case should be more able to meet those specifications than the original ’best case’.
- data of the Behaviour and Structure class which are strongly related to those design requirements (of the Function class) which are indeed met by the ’best case’

  By adding these data to the target-file it is tried to find cases which not only meet the unsatisfied specifications, but also resemble the ’best case’. When considering the Behaviour and Structure data as the solution to the design problem which is expressed by the Function data, matching these data leads to adaptation cases which already contain much of the solution which has been suggested by the ’best case’. The focus is on those parts of the solution which can be related to the ’solved’ design problems.

> So, from the cases which meet the unsatisfied specifications (the first type of data), those cases are preferred with the most resemblance with the ’best case’. The idea is that this will lead to a more realistic ’adapted case’, which is a combination of the ’best case’ and the ’adaptation case’.

To find the unsatisfied specification data is straightforward: examine the results of the first phase. From Figure 5.7 it is read that the take-off and landing distances specifications are not met by the Fokker 70. So the target-object TOLanding is copied into the adaptation target-file.

To be able to choose the other type of data, which ensures a resemblance of the cases with the ’best case’, expert knowledge is required. Next table shows some general relations between the common specification categories (of the class Function) and the related data of the classes Behaviour and Structure.

Table 5.6. Relations between typical specification data and other main aircraft data.

Function class	Behaviour class	Structure class
passenger/ payload - range	max. take-off weight (MTOW) ratio payload/ MTOW ratio fuelload/ MTOW	engine type fuselage lay-out ratio fuselage width/ length
max. speed	thrust or power loading	engine type wing sweep wing thickness
take-off and landing distances	wing-loading thrust- or power-loading	number of engines
ceiling	wing-loading thrust- or power-loading	engine type fuselage cross section

Table 5.6 gives just general relationships. The designer may use different relationships; this is possible because these relationships have not been automated.

This results in the adaptation target-file which is shown in Figure 5.10.

```
OBJECT TOLanding PRIO 100
TOLand.to_length               1          1400
TOLand.l_length                1          1200

OBJECT Configuration PRIO 33
Airplane_S.engine_type         2          turbofan
Airplane_S.nr_of_engines       5          2
Airplane_S.fus_cross_type      2          circle

OBJECT Weight PRIO 33
Airplane_W.MTOW                1          39.92
Airplane_W.fuelload_ratio      1          0.28000

OBJECT Wing PRIO 33
Wing_S.sweep                   1          18
Wing_S.thickness_root          1          0.123
```

Figure 5.10. The adaptation target-file.

The next remarks can be made:

- The target-object which is related to the unsatisfied specifications has the highest priority-number of 100. The other three target-objects have lower priority-numbers to express their lesser importance. Together they are as important as the first target-object: the sum of their priority-numbers equals 100.
- The Behaviour and Structure data are organized according to the classification structure. This structure makes is possible to show the similarities per data class, which makes it easy to examine the relevant data per aircraft component or aspect (i.e. Structure resp. Behaviour).
- The value of parameter Airplane_W.fuelload_ratio has five digits. This is to indicate that this value has been estimated when the case was created, not because the accuracy is so high. (For the Fokker 70, the estimation was based upon the fuel volume.)

5.4.3 Output of the CBR-module: the similarity scores

The results of the matching process with the adaptation target-file of Figure 5.10 is shown in Figure 5.11. Only the best 16 cases have been listed.

Case	Global	TOLanding	Configuration	Weight	Wing
Target prio. →		100	33	33	33*
BAe RJ70	95	100	67	100	100
BAe RJ85	95	100	67	100	100
Fokker 70	75	50	100	100	100
Dornier 328-110	70	100	67	50	0*
BAe RJ100	70	50	67	100	100
ATR 72-200	65	100	34	50	0*
Saab 2000	62	100	67	0	0*
Embraer 145	59	50	100	50	50*
Fokker 100	50	0	100	100	100
MD C17	45	50	67	0	50*
B 737-400	42	0	100	100	50*
B 737-300	42	0	100	100	50*
MD 87	36	0	67	100	50*
MD 90-30	36	0	67	100	50*
A 320-200	36	0	67	100	50*
A 321	36	0	67	100	50*

Figure 5.11. The results of the similarity measurements between the target-file and the cases. (: thickness ratio-data missing) (grey rows: better local score, 100%, than the 'best case', the Fokker 70: 50%)*

Some general remarks can be made:

- The best matching results are given by almost the same set of aircraft as the results of the original target-file. Only the Airbus 320-200 has not been shown in Figure 5.7; and the Boeing 737-500 and the Canadair RJ200 did appear in Figure 5.7, but not in Figure 5.11.
- Only two aircraft show better global similarities than the selected 'best case', i.e. the Fokker 70. These are the British Aerospace RJ70 and RJ85 aircraft.
- Five aircraft show better similarity scores on the take-off and landing distances: the BAe RJ70 and RJ85, the Dornier 328, the ATR 72 and the Saab 2000. Three aircraft show identical similarity scores on that target-object: the BAe RJ100, the Embraer 145 and the MD C17. Because the other aircraft have even worse take-off and landing distance performances (too short or too long), they do not seem capable to improve the selected 'best case'. Therefore they will not be considered again.

- When the similarity scores are examined per target-parameter, it is seen that the 50% scores on the target-object TOLanding is the result of similar landing distance domains, and different take-off distances. This score distribution is identical to the Fokker 70 results.

  Therefore it is not expected that these cases will improve the ’best case’, and consequently they will not be used for the adaptation process.
- The ATR 72-200 scores locally only 34% on the Configuration target-object. This is because the aircraft has turboprop engines, and because its fuselage cross-section is a double-bubble (flattened underside, such as the Fokker 50) instead of a circle. This difference may not be that important, but with this target-file (Figure 5.10) it appears to be as important as the type of engine.

So, only five cases are left for the adaptation process: the BAe RJ70 and RJ85, the Dornier 328, the ATR 72 and the Saab 2000; these have been given a grey colour in the figure. In next section it is discussed which of these ’most promising adaptation’ cases is selected.

5.4.4 Selecting the adaptations

In next figure the configurations of the five ’most promising adaptation’ aircraft are shown, together with the configuration of the ’best case’, the Fokker 70. These aircraft do better match the take-off and landing distance specifications than the Fokker 70.

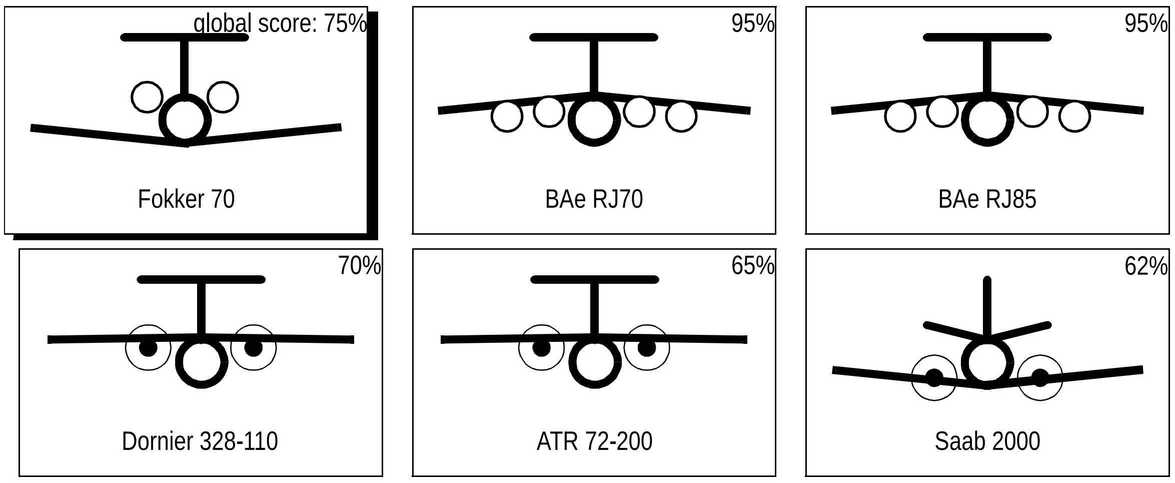


Figure 5.12. The configurations of the ’most promising adaptation’ cases.

Proposals for adaptations are based on the examination of these configurations:
- there are no aircraft with exactly the same configuration as the ’best case’;
- the configuration with engines mounted on high wings and with T-tails seems favourable;

- from the presented configurations it is difficult to distinguish a preference for the engine type;
- also a configuration with four engines seems to be a viable option.

With these remarks, the BAe RJ70 is preferred as a source from which to copy the adaptation data. The next configuration components are selected for copying:

- the high wing configuration;
  because this configuration often appears in Figure 5.12;
- the engines mounted on the wings;
  because this comes with the high wing configuration.

The type of engine is not changed, neither do the number of engines and the type of tail. This results in a novel configuration, since an aircraft with two turbofans mounted on high wings is not found in the case-base.

The T-tail and the turbofan are left unchanged because the BAe RJ70 does have an identical type of tail and the same type of engines. The number of engines has not been changed because it is believed that the other adaptations could be enough, in first instance.

5.4.5 Generating the adapted case

So it is decided to adapt the ’best case’, the Fokker 70, with the high wing configuration and the engine location of the BAe RJ70. The result is sketched in Figure 5.13:

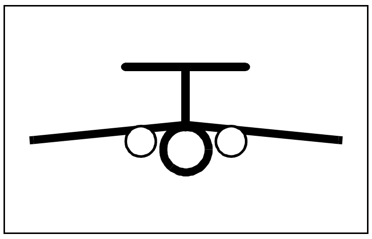


Figure 5.13. The new, adapted configuration.

This change in the configuration, however, also has effect on other aircraft data. For example some Structure data has to be copied from the BAe RJ70 to be able to make a rough but realistic three-dimensional model of the adapted aircraft, which is performed by the Geometrical Module (Section 5.6). And some Behaviour data will also be affected, which is used in the Functional module (Section 5.5). Next data is affected by the adaptation of the wing and engine configurations:

- the longitudinal location of the wings
  Because the engines move forward from the aft-fuselage to the wings, the centre of gravity of the aircraft will move forward. For stability reasons the wings have to be relocated too, more forward.

The non-dimensional parameter used for this purpose is Wing_S.MAC_loc, which is defined as the location of the 1/4-chord location of the MAC (Mean Aerodynamic Chord) with respect to the centre of the fuselage, relative to the MAC.

- the attitude of the wings
  For stability reasons, the dihedral (angle between horizontal plane and the wing) is different for a high wing and a low wing: typically a high has a negative dihedral and a low wing a positive. The incidence (angle between vertical and longitudinal plane and the wings) is not changed.
- the shape and size of the wings
  Since wing-mounted engines have effect on the structure and the aerodynamics of the wings, their shape will have to be reconsidered. Therefore all data that describe the wing of the BAE RJ70 is copied, including the shape parameters such as the aspect ratio A, the taper, the sweep and the thickness ratio.

  Because the wing parameters that have a clear effect on the specified performances obviously do not differ very much (since the local matching score of the adaptation object-target Wing of the BAe RJ70 is 100% in Figure 5.11), it is expected that these adaptations will not affect those performances that much.

  By copying all wingdata of the BAe RJ70, the overall wingsize is also changed, such as the wing gross-area S. Because the weight parameters of both original cases differ distinctively, the wing-loading W/S of the new adapted case will have to be reconsidered. However, a realistic value can not be copied from the two original cases, since their wing-loading values are based on different combinations of weights and wing areas.

  Therefore, the resulted combination of the ’best case’ and the ’best adaptation case’ will contain inconsistent data. This is inherent to this technique, however, as has been explained before in Section 4.3.4. One of the purposes of the next module, the Functional Module, is to overcome these inconsistencies.
- the spanwise location of the engines
  Because the engines will be mounted on the wings instead of the rear fuselage, the original spanwise location of the engines of the Fokker 70 are not realistic. However, the BAe RJ70 has four engines. The question than rises: which spanwise location to copy? It is decided to choose the inboard locations, which seems to ’look better’ than the outboard locations; also the asymmetry will be less when one engine is inoperative.

The resulting combined case is saved in a separate file, in the same format as the other case-files. So it is readable by the Geometrical Module.

5.4.6 Discussion on the use of the CBR-module

The matching process with combined parameters

The parameters 'Range' and 'Payload' (or 'Number of passengers') have to be considered in combination with each other. Because their influence on each other is so big, they can not be evaluated separately. However, their domains are defined independently. The problems of this approach is explained with next figure.

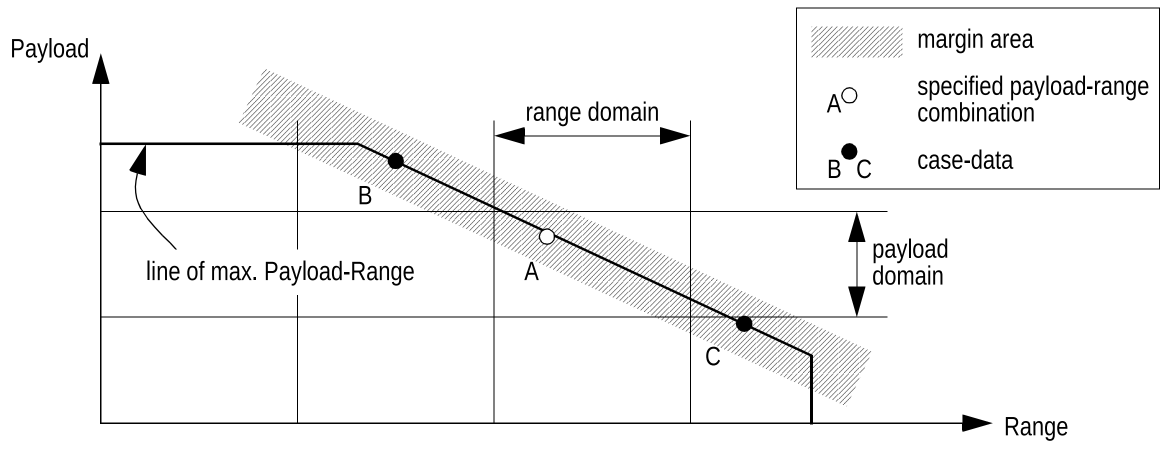


Figure 5.14. The payload-range diagram of one aircraft.

Figure 5.14 represents the payload-range performance of one aircraft. Suppose points B and C represent two payload-range combinations that are known and stored in the case-base, and point A represents a specified payload-range combination. Because point A lies well on the performance line of the aircraft (line of max. Payload-Range), the aircraft should perfectly match this payload-range specification. However, because points B and C neither share a parameter domain with point A, the result of the similarity measurement will be 0% on these parameters. Hence the aircraft will show a lower similarity score than it should do.

To overcome this problem the CBR-tool should measure the similarity with the margin area indicated in Figure 5.14, instead of the individual domains of the two parameters. This could be possible by defining a separate parameter which is a function of the 'Range' and the 'Payload' parameters.

A practical problem to this theoretical solution is to determine the payload-range diagram of the aircraft, or at least the gradient of the sloping line. You need at least two payload-range combinations per aircraft, which are often not available in the literature.

The binary matching criteria

The matching process of the Cbr-tool has only two possible results: the parameter matches or it does not match. It is obvious that this binary matching criterion is not flexible.

Another approach could be to express the similarity of two parameter values by their difference, absolute (δx) or relatively ($\delta x/x$). The currently used CBR-tool is not able to do this, however, because of its internal network representation of the similarities.

Relative similarities would only be useful for continuous parameters. A major disadvantage of using these relative similarities is that this would require an extra calculation step between the retrieval of the data values and the matching process.

Another disadvantage would be that some physically based domain boundaries would be lost.

Implicit relations between specifications and the configuration

One of the conclusions about the local similarities between the target-object TOLdistance and the configuration, see Table 5.4, is that aircraft with turbofan engines mounted on the wing generally need too long take-off and landing distances.

This can be understood by applying domain knowledge of aircraft design. Generally, larger or heavier aircraft need longer take-off and landing distances. Although heavier aircraft need larger engines, the relative size of their engines is small compared with the engine size of smaller aircraft. Therefore the larger aircraft have less space problems for mounting the engines under their low wings, than the smaller aircraft. So there is a correlation between the size of the aircraft, determined by the number of passengers, and the engine configuration: small aircraft do seldom apply the wing-mounted turbofan engine configuration, unless the aircraft has a high-wing configuration.

So the conclusion on itself is not correct, but it is true for small take-off and landing distances which are often related with small aircraft.

There are also other arguments for the choice of the engine configuration, which are not performance driven. For example the accessibility for the maintenance, and the availability of engines at the design date.

The success of the existing aircraft is also not considered. The sale numbers could be an indication for that, but other arguments may be at least as important, such as political and marketing issues.

Why the Saab 2000 with the highest possible similarity score has not been chosen

Although the Saab 2000 has the highest possible similarity score, the aircraft has not been chosen as the ’best case’. Instead the Fokker 70 has been chosen; see Section 5.3.5. There are some reasons for that decision which have not been mentioned in Section 5.3.5, and which have nothing to do with proper designing.

- Detailed engine information was available for the BMW-RR BR710 turbofan in [Fransen, 1996]. Because this data is needed by the calculations with the Functional module, that type of engine was favourable.
- Because the Saab 2000 has the highest possible similarity score, no adaptation should be needed when it is chosen as the ’best case’. However, the purpose of this chapter is to examine the proposed strategy and AI techniques, including the adaptation phase.

5.5 First modifications with the Functional module

5.5.1 Introduction

With the Case-Based Reasoning module a proper starting point for the design process has been produced: an aircraft concept was chosen, as well as initial values of some important parameters. These values are copied from existing cases, however, and need to be reconsidered for the current design problem.

This reconsideration task is supported by the Functional module. The module applies the tool QUAESTOR [van Hees, 1997] to evaluate the chosen solution and to see whether the specified Function parameters are met. QUAESTOR builds a network of relations and equations that connect the Structure parameters with the Function parameters; see Section 4.4. The designer uses the link to perform sensitivity studies in order to choose appropriate parameter values.

The building of the network of relations is an interactive process: QUAESTOR suggests appropriate relations, and the user selects them. QUAESTOR manages all the background activities for the construction of the network.

In Section 4.4 it was described how such a numerical network can be constructed with QUAESTOR. This section shows how the network is built and used in this particular case.

Section 5.5.2 deals with the development of the knowledge base in QUAESTOR, which has to be completed before the tool can be applied. Note that the QUAESTOR terminology is used which differ from typical optimisation terminology. Therefore a different font is applied for those QUAESTOR terms, such as `relations`, `constraints` and `parameters`.

The next section, Section 5.5.3, describes the actual building of the network; called a `template` in QUAESTOR terminology.

When the `template` has been created, the user defines a set of values to perform the sensitivity studies. This is described in Section 5.5.4. In the next section, Section 5.5.5, it is explained which values are chosen, based on the sensitivity studies.

In the last section this procedure is evaluated and conclusions are drawn: Section 5.5.6.

5.5.2 The pre-processing work: building the knowledge base

Before the tool QUAESTOR can be used to build a numerical network, the elements of which the network consists have to be defined. These elements are the `relations`, `constraints` and the `parameters`, and are collected in the knowledge base of QUAESTOR. Mind that the meaning of these terms differ from typical optimisation terminology:

- the `relations` express the numerical knowledge in equation-format;
- the `constraints` are linked with `relations` to limit their validity; they are evaluated to be ’satisfied’ or ’unsatisfied’, and can be expressed by equalities as well as inequalities;
- with the `parameters` every variable, constant, parameter or another quantity is represented.

The elements are stored in `frames.` They represent the knowledge as can be found in aircraft preliminary design literature.

`relations`

Numerical relations are put into the knowledge base in the format
parX=f(*parY1*, *parY2*, .., *parYn*);

One `parameter` is in the left-hand clause of the equality, and a function with zero, one or more `parameters` is in the right-hand clause. For example the expression of the cruising speed:

$$ToverW = \frac{W_{ratio}}{T_{ratio}} \cdot \left[q \cdot \frac{C_{D0}}{WoverS} + \frac{WoverS}{q \cdot \pi \cdot A_{wing} \cdot e} \right] \qquad \text{(Eq. 5.1)}$$

This equation can be put straightforward into the knowledge base; next figure shows the representation of the `relation` in development mode of QUAESTOR.

```
Relation   < TowerW=W_ratio/T_ratio*(q*CD0/WoverS+WoverS/(q*pi*A_wing*e))  |
Control    < HARD, TW, AND, ON, EMPIRICAL, CLASS: Undefined!                >
Reference  < Min. thrust-loading for ’cruising speed’ condition             |
Data       <                                                                |
Constraint < 1                                                              |
```

Figure 5.15. A `relation frame` in Quaestor.

Each `relation` has five slots:

- a Relation slot
  This slot contains the numerical expression of the `relation`. A wide variety of functions are available, such as geometric functions, interpolation functions, logical functions, etc.. There are also functions which call for external data or programs.
- a Control slot
  This slot contains attributes which define how the `relation` can be used. These attributes directly affect the selection process for suitable `relations`, and some of them are useful during the development process. Some important attributes are:
  - OW or TW (i.e. ’One Way’ or ’Two Way’)
    ’One Way’ means that the `relation` will only be used to calculate the `parameter` in the left clause of the `relation`; the `relation` is considered as a *function*. This is particularly meaningful for empirical relations, which should not be used to determine a `parameter` in the right clause.
    ’Two Way’ means that the `relation` can be considered as an *equation*, which can be served to calculate unknown `parameters` in the right as well as in the left clause.
  - AND or OR
    This attribute defines whether all (AND) or just one (OR) of the connected `constraints` have to be satisfied in order to authorise the use of the `relation`.

  The ">"-sign at the right side of the rule indicated that more information may be available.
- a Reference slot
  In this slot background information is given, which may be as large as several pages. This is very helpful when the knowledge base grows larger.
- a Data slot
  This slot may contain data which is needed by functions that are applied in the `relation`.
- a Constraint slot
  This slot contains the number of `constraints` which are linked with the `relation`.

`constraints`

When the applicability of a `relation` is limited, this can be expressed by linking one or more `constraints` to it. Only when one or all linked `constraints` are satisfied (depending on the control attributes of the `relation`), the `relation` is nominated for the construction of the relational network, i.e. the `template`. A `constraint` may be linked with more `relations`.

The format of a `constraint` is less strict then the `relation`. The left clause can also be a function with more `parameters`, and the operand may be any kind of logical operator:

f(*parX1*, *parX2*, .., *parXn*) | =,<,OR etc. | g(*parY1*, *parY2*, .., *parYn*).

An example is the `constraint` which is linked with the `relation` for the cruising speed condition, Figure 5.15, indicating that the `relation` is only valid for "cruising speed" conditions. The `frame` for that `constraint` is shown in next figure:

```
Constraint < Flight_phase="cr"                                   |
Control    < EPS 0.5, HARD, TR, ON, NO MESSAGE                   |
Reference  < cruising                                            |
Data       <                                                     |
```

Figure 5.16. A `constraint frame` in QUAESTOR.

The `constraint frame` looks like the `relation frame`, and contains four slots:

- a Constraint slot
  This slot contains the expression of the `constraint`. As with the `relation` slot, many functions are available. Notice that also string-operations are possible.
- a Control slot
  The control attributes in this slot influence the use of the `constraint` and help the development of the knowledge base. These control attributes differ from those of the `relation frame`. Some of the most important are:
  - HARD or SOFT
    When this attribute is set to SOFT and the `constraints` is evaluated to be FALSE, the user is asked whether the linked `relation` can be used or not. When the attribute is set to HARD, QUAESTOR will automatically reject the linked `relation` from the `template`.
  - TR or FA (i.e. TRUE or FALSE)
    The attribute value FALSE stands for a logical NOT, meaning that when the expression in the `constraint` slot is FALSE, the `constraint` is regarded as TRUE. This is convenient when the formulation of the `constraint` is very complex whereas the negation can be written as a much simpler expression.
- a Reference slot
  In this slot background information is stored.
- a Data slot
  Also in this slot data can be stored which is needed by functions in the Constraint slot.

parameters

Each time a new `relation` or `constraint` is defined in the knowledge base, QUAESTOR links it with all involved `parameters`. When a `parameter` has not been defined before, QUAESTOR prompts the knowledge base developer to do so.

Next figure shows the frame of the `parameter` *Flight_phase*:

```
Parameter  < Flight_phase                                                    |
Control    < COL 9, VR, OUT, STRING                                          |
Reference  < flight condition; pick list:                                    >
Data       <                                                                 |
Value QWB  <                                                                 >
Dimension  <                                                                 >
Class      < Undefined!                                                      |
```

Figure 5.17. A `parameter frame` in QUAESTOR.

The `parameter` frame consists of seven slots:

- a Parameter slot
  This slot contains the symbol of the `parameter`. The ">"-sign at the right site of the rule indicates that more information may be available.
- a Control slot
  The attributes in this slot affect the use of the `parameters`. Again, these attributes differ from the Control attributes of the `relation` and `constraint frames`. Some important attributes are:
  - INI <value> (only available for numerical `parameters`)
    This attribute defines its initial value, which is necessary for the Newton-Raphson and Simplex methods to solve iteration loops in the `template`.
  - USR/USL or SYS/SYL or VR
    This attribute suggests how the `parameter` will be used: its value is possibly input by the user (USR/USL), never input by the user but determined by other `relations` (SYS/SYL), or its value is always input by the user (VR). The USR and SYS values suggest this for the right clause, while USL and SYL suggest this is valid for the left clause.
  - NN or N or NP or P or NZ or U (only available for numerical `parameters`)
    This attribute controls which value the `parameter` can get: Non Negative (NN), Negative (N), Non Positive (NP), Positive (P), Non Zero (NZ) or Unrestricted (U). When the calculated value is outside the defined range, QUAESTOR produces a warning.
  - VALUE or STRING or OBJECT
    This attribute defines the type of `parameter`: numerical (VALUE), a string (STRING) or a QUAESTOR-object (OBJECT). A QUAESTOR-object consists of a multi-dimensional table filled with values. Such an object is for example applied to implement the engine thrust curves of an engine (maximum thrust

values for several temperatures and altitudes). Specialised functions are available for the `relations` to read these tables and interpolate between the table values.

- a Reference slot
  In this slot the `parameter` is described in words.
  It is also possible to put a so-called picklist to this slot, such as in Figure 5.18. A picklist contains values from which the user can select one. This is very helpful when it is important to use specific string-values, as is the case with the `parameter` Flight_phase. Another useful way of a picklist is when empirical `parameters` are applied which can adopt only pre-defined values.
  The picklist is activated by moving to the right column, indicated by the ">"-sign.

```
flight condition; pick list:

"cOEI" <EQ> climbing with One Engine Inoperative
"toAE" <EQ> take-off with All Engines operative
"sscOEI" <EQ> second segment climb with One Engine Inoperative
"cr" <EQ> cruising
"land" <EQ> landing
```

Figure 5.18. The picklist of the `parameter` *"Flight_phase" (see Figure 5.36).*

- a Data slot
  This slot can also contain a table with values. The difference with the Reference slot is that when a function in an expression needs table-values, only the Data slot will be consulted.
- a Value slot (`QWB` stand for the QUAESTOR Work-Base)
  This slot shows the value of the `parameter`, as it is registered in the QUAESTOR Work-Base. When the value has not been determined yet, the slot is left blanc.
- a Dimension slot
  In this slot the dimension of the `parameter` can be described.
- a Class slot
  Each `parameter` can be categorised into a class, which is indicated in this slot. The classification structure helps the developer or user to find a specific `parameter`.

There are several possibilities to examine and browse the knowledge base. One way is to scroll through the `frames`, as described here. Other options are the Parameter List, which shows the `parameters` per Parameter Class, and the Expression List, showing all the expressions (`relations` and `constraints`).

All `frames` can be saved in QUAESTOR knowledge-base files.

5.5.3 Building the `templates`

When the developer has filled the knowledge base, QUAESTOR builds the template in close interaction with the user.

The user starts by selecting a `parameter` which value has to be calculated. This is called a goal `parameter`. QUAESTOR builds the template by searching for `relations` that can derive the goal `parameter` from other `parameters`. It works its way backwards until only independent `parameters` are left; the so-called problem-driven strategy.

In order to construct the template, QUAESTOR applies a simple inference algorithm which is displayed in Figure 5.19 and discussed next:

```
define GOAL parameter
determine GOAL value
    find appropriate RELATION
    if CONSTRAINT is linked
        provide CONSTRAINT parameter values
        if value is "not known" (not derived, no input)
            (sub)GOAL = CONSTRAINT parameter
            determine (sub)GOAL value
    if no CONSTRAINT is linked or linked CONSTRAINT is "Fulfilled"
        suggest the RELATION
        if RELATION is "accepted"
            provide RELATION parameter values
            if value is "not known" (not derived, no input)
                (sub)GOAL = RELATION parameter
                determine (sub)GOAL value
    calculate GOAL parameter value
construction of template is completed
```

Figure 5.19. The inference algorithm of QUAESTOR for the construction of a template.

The interaction with QUAESTOR starts by selecting a knowledge-base file. The user chooses the top-goal parameter from the knowledge-base, i.e. the (goal) parameter for which the template is constructed. When the user has chosen the top-goal `parameter`, QUAESTOR searches for `relations` which contain that `parameter`; whether the `parameter` is in the left-hand clause or in the right-hand clause of an equation. The user selects one of the suggested `relations`, and QUAESTOR inquires each involved `parameter`. When the `parameter` has not been derived yet, the user is asked for a value or to define it as a new goal `parameter`. In the latter case, QUAESTOR applies the inference algorithm again. When all `parameters` have been labelled either ’dependent’ (derived by the template) or ’independent’ (input by the user), the inference will stop and the template is complete.

Before an appropriate `relation` is suggested to the user, linked `constraints` are examined. When the involved `parameters` are not known to the workbase of QUAESTOR or to the user, QUAESTOR applies the inference algorithm to derive its value. Only when all linked `constraints` have been validated to be "fulfilled", the `relation` will be suggested to the user.

QUAESTOR is able to detect cycles or systems of more then one equation. With (pre-defined) initial values and the Newton-Raphson and Simplex algorithms the `parameters` are approximated (with a pre-defined accuracy).

The Newton-Raphson and Simplex algorithms are also used to determine the value of the top-goal `parameter`. The template is internally translated into a set of equations, with the number of independent `parameters` equal to the number of `relations`. Because of this strategy it does not matter whether the goal `parameters` are in the left-hand clause or the right-hand clause of the equations.

From his experience the designer knows that the wing-loading-thrust-loading diagrams are important to express the influence of important design variables on the aircraft performances. In our study case such a diagram is constructed to evaluate some of the design variables copied from the (adapted) case, with respect to the design specifications. Five relational networks are constructed with the aid of QUAESTOR, representing five design specifications:

- cruising speed specification;
- take-off distance specification;
- landing distance specification;
- en-route climbing with one-engine inoperative (OEI) specification (specified by the FAR regulations);
- second segment climb with one engine inoperative (OEI) specification (a FAR regulation).

At this stage the range specification is not taken into account. This requires too many details about the aircraft weight distribution. Instead, the maximum take-off weight is considered as fixed and is copied from the data-file of the Fokker 70, the 'best-matching' case of Section 5.3.

The next example shows the working of and the interaction with QUAESTOR. In the example a template is constructed for the calculation of the required thrust-loading to satisfy the cruising speed specification:

- The user/designer retrieves the appropriate knowledge-base file.
  The concerned file contains the `relations`, `constraints` and `parameters` which may be used for all specifications.
- The user selects the goal `parameter`: `ToverW.`
- QUAESTOR searches for `relations` which contain the `parameter ToverW.`

There are several `relations` with the `parameter` ToverW. The first `relation` found by QUAESTOR is one of the wing-loading-thrust-loading `relations` representing one of the six design specifications. All these `relations` are linked with a `constraint` about the considered flight-phase. Before the `relation` can be suggested by QUAESTOR, the `constraint` is evaluated.

- QUAESTOR detects a `constraint` linked to the `relation` and evaluates it.
  The expression for this `constraint` is for example: `Flight_phase="TO_AE"`.
- QUAESTOR asks for the value of the `parameter` `Flight_phase`.
  To be able to evaluate the `constraint`, the value of the `parameter` `Flight_phase` has to be known.
- The user selects the value `"CR"` from the picklist of the `parameter` `Flight_phase`.
  The `parameter` Flight_phase is a string `parameter`. To be able to effectively use the `parameter` in a `constraint`, the user is prompted to choose a string-value from a pre-defined set. He can choose between: `"TO_AE"` (Take-Off condition with All Engines running), `"SSC_OEI"` (Second Segment Climb condition with One Engine Inoperative), `"C_OEI"` (Climbing condition with One Engine Inoperative), `"CR"` (CRuising condition).
- QUAESTOR certifies that the `constraint` `Flight_phase="CR"` is fulfilled, and the others are not fulfilled.
  Because the `constraint` `Flight_phase="TO_AE"` is found to be unfulfilled, the linked `relation` is not suggested. The next possibly suitable `relation` is for example the wing-loading-thrust-loading `relation` which is linked with the `constraint` `Flight_phase="SSC_OEI"`. That `constraint` is evaluated and ascertained to be unfulfilled, so QUAESTOR will look for another suitable `relation`. This process continuous until the wing-loading-thrust-loading `relation` is found which is linked with the `constraint` `Flight-Phase="CR"`.
- QUAESTOR suggests the wing-loading-thrust-loading `relation` which expresses the cruising speed specification.
  The suggested `relation` is the expression of Equation 5.1.
- QUAESTOR asks for the user to provide values for the new `parameters`.
  QUAESTOR evaluates each `parameter` in the `relation`: whether it has already been determined by other `relations`, if it has a pre-defined value (i.e. is a constant), or if it should be calculated by other `relations` (as indicated by the control attributes of the `parameter`). When all these checks are negative, the user is asked to provide their values.
- QUAESTOR assigns new goal `parameters`.
  When the user does not provide a value when requested, that `parameter` is supposed to be determined by other `relations`. QUAESTOR assigns those `parameters` to be new goal `parameters`.
- QUAESTOR introduces the `relation` into the `template`.

When all `parameters` in the `relation` have been labelled as ’derived’ (calculated by `template`), ’input’ (provided by user or by external file) or ’requested/pending/unknown’ (goal), the `relation` is accepted and introduced into the `template`.

- QUAESTOR evaluates the still incomplete `template`.
  QUAESTOR reviews the incomplete `template`. It examines the overall situation by counting the number of accepted `relations` and the number of unknown `parameters`, and it checks for local loops. When a loop is detected, QUAESTOR uses Newton-Raphson and Simplex algorithms to determine the values of the involved `parameters`.

  At this stadium QUAESTOR counts 1 accepted `relation`, i.e. the expression of Equation 5.1, and 4 unknown `parameters`: `T_ratio`, `q`, `CD0` and the top-goal `parameter` `ToverW`. Because there are still more unknown `parameters` than accepted `relations`, the top-goal `parameter` can not be calculated and QUAESTOR continues.

  Note that the linked `constraints` are not considered in this respect, since they do not produce new `parameter` values.
- QUAESTOR assigns a new goal `parameter`: `CD0`.
  All three new unknown `parameters` will be goal `parameters`. QUAESTOR randomly focuses on one of them: `CD0`. The other `parameters` will be dealt with later on.
- QUAESTOR suggests a `relation` to calculate `CD0`.
- Etc.

The complete inference process is displayed in Figure 5.20. The previous text covers only the first couple of inference steps.

With respect to this specific inference process, some typical difficulties are revealed:

- The user has to make assumptions about `parameters`.
  For instance the `parameter` `Wto` is chosen to be constant, instead of the `parameter` `S_wing`. Because the expression `WoverS=Wto/S_wing` is used in the template, the value of `S_wing` will change when `WoverS` is varied.

  As a consequence it is possible to make the `parameter` `Tto` variable and calculate it by the expression `ToverW=Tto/Wto`.

  This issue of deciding which `parameter` to keep constant and which to vary, is not a specific QUAESTOR-problem; it is typical for the creation of such a template.
- The user has to have a good overview on the `relations` available.

It is too complex to draw a scheme of the complete `template`. Figure 5.21 shows the 19 `relations` and the 19 dependent `parameters`, i.e. the `parameters` that are calculated within the `template`.

```
START OF INFERENCE:
  ToverW is TOPGOAL and chains to:
    ToverW=f(W_ratio,T_ratio,q,CD0,WoverS,pi,A_wing,e)
    CD0 is SUBGOAL of ToverW and chains to:
      CD0=f(r_Re,r_uc,CD0S_t,CD0S_w,CD0S_f,CD0S_n,S_wing)
      S_wing is SUBGOAL of CD0, ToverW and chains to:
        WoverS=f(Wto,S_wing)
      S_wing inferred
      CD0S_n is SUBGOAL of CD0, ToverW and chains to:
        CD0S_n=f(r_n,r_thr,bypass_ratio,Tto,T_to_spec,p_0)
        Tto is SUBGOAL of CD0S_n, CD0, ToverW and chains to:
          ToverW=f(Tto,Wto)
        Tto inferred
      CD0S_n inferred
      CD0S_f is SUBGOAL of CD0, ToverW and chains to:
        CD0S_f=f(r_f,l_f,b_f,h_f)
      CD0S_f inferred
      CD0S_w is SUBGOAL of CD0, ToverW and chains to:
        CD0S_w=f(r_w,toverc_mean,sweep,pi,S_wing)
      CD0S_w inferred
      CD0S_t is SUBGOAL of CD0, ToverW and chains to:
        CD0S_t=f(r_t,CD0S_f,CD0S_w)
      CD0S_t inferred
      r_Re is SUBGOAL of CD0, ToverW and chains to:
        r_Re=f(Re_f)
        Re_f is SUBGOAL of r_Re, CD0, ToverW and chains to:
          Re_f=f(V,l_f,kin_visc)
          kin_visc is SUBGOAL of Re_f, r_Re, CD0, ToverW and chains to:
            kin_visc=f(temp,rho_atm)
            rho_atm is SUBGOAL of kin_visc, Re_f, r_Re, CD0, ToverW and chains to:
              rho_atm=f(rho_0,temp_ratio)
              temp_ratio is SUBGOAL of rho_atm, kin_visc, Re_f, r_Re, CD0, ToverW
                                                                       and chains to:
                temp_ratio=f(h,temp_0)
              temp_ratio inferred
            rho_atm inferred
            temp is SUBGOAL of kin_visc, Re_f, r_Re, CD0, ToverW and chains to:
              temp=f(temp_ratio,temp_0)
            temp inferred
          kin_visc inferred
          V is SUBGOAL of Re_f, r_Re, CD0, ToverW and chains to:
            M=f(V,a)
            a is SUBGOAL of V, Re_f, r_Re, CD0, ToverW and chains to:
              a=f(gamma,R,temp_0,temp_ratio)
            a inferred
          V inferred
        Re_f inferred
      r_Re inferred
    CD0 inferred
    q is SUBGOAL of ToverW and chains to:
      q=f(rho_atm,V)
    q inferred
    T_ratio is SUBGOAL of ToverW and chains to:
      T_ratio=f(alt,M)
      alt is SUBGOAL of T_ratio, ToverW and chains to:
        alt=f(h)
      alt inferred
    T_ratio inferred
  ToverW inferred
END OF INFERENCE
```

Figure 5.20. The inference process of QUAESTOR for the creation of the cruising specification template.

```
Contents of current template: 19 parameter(s) and 19 expression(s)

     CD0S_f   zero-lift drag area due to fuselage ..........[m^2]
     ToverW   thrust-loading ................................[-]:        ?
     T_ratio  actual thrust ratio (T/Tto) ...................[-]
     q        dynamic pressure ...........................[N/m^2]
     CD0      zero-lift drag coefficient; picklist; ..........[-]
     r_Re     zero-lift drag corr. factor for Reynolds numb ..[-]
     CD0S_t   zero-lift drag area due to tail ..............[m^2]
     CD0S_w   zero-lift drag area due to wing ..............[m^2]
     CD0S_n   zero-lift drag area due to engine and nacelles[m^2]
     S_wing   wing surface .................................[m^2]
     Tto      static take-off thrust .........................[N]
     Re_f     Reynolds number of fuselage; picklist: .........[-]
     V        velocity ....................................[m/s]
     kin_visc kinematic viscosity ........................[m^2/s]
     temp      ...............................................[K]
     rho_atm  Atmospheric density .......................[kg/m^3]
     temp_ratio
            Relative atmospheric temperature ...............[-]
     a        Speed of sound ..............................[m/s]
     alt      Altitude in [ft] .............................[ft]

     CD0S_f=0.0031*r_f*l_f*(b_f+h_f)
     ToverW={W_ratio}/{T_ratio}*({q}*{CD0}/WoverS+WoverS/
                                                    ({q}*{pi}*{A_wing}*{e}))
     T_ratio=LININT(4,3, "alt","Mach","T_ratio", alt,M)
     q=.5*rho_atm*V^2
     CD0=r_Re*r_uc*(CD0S_t+CD0S_w+CD0S_f+CD0S_n)/S_wing
     r_Re=47.0/Re_f^(0.2)
     CD0S_t=(r_t-1.0)*(CD0S_f+CD0S_w)
     CD0S_w=0.0054*r_w*(1+3*toverc_mean*(COS(sweep*pi/180.))^2)*S_wing
     CD0S_n=1.72*r_n*r_thr*(5+bypass_ratio)/(1+bypass_ratio)*Tto/(T_to_spec*p_0)
     WoverS=Wto/S_wing
     ToverW=Tto/Wto
     Re_f=V*l_f/kin_visc
     M=V/a
     kin_visc=1.458E-06*temp^1.5/((temp+110.4)*rho_atm)
     temp=temp_ratio*temp_0
     rho_atm=rho_0*temp_ratio^4.2558761
     temp_ratio=1.0-0.0065*h/temp_0
     a=SQRT(gamma*R*temp_0*temp_ratio)
     alt=h/0.3045
```

Figure 5.21. The 19 `relations` and 19 dependent `parameters` that makes the template solvable.

The next figure, Figure 5.22, presents the independent `parameters`, which values are not derived within the `template`. Some of the `parameters` are physical constants, and their values have been given by the knowledge base developer and saved within the knowledge base. These `parameters` are of the Constant Class.

Other values are input by the user. They come from the adapted case of the previous design step, or their values are suggested by the picklists.

Another source of information is the thrust-ratio versus speed diagram of the engine. In the similar AGARD study, see [Fransen, 1996], such a diagram is presented for a specific engine, the Rolls-Royce BMW RB512 turbofan. The non-dimensional data of this engine is put into a QUAESTOR `relation` via interpolation functions and data tables in the data slot. By using this data the engine is scaled, also known as the rubberizing process. When such diagrams are available for other engines, the user can choose between different engines. The selection of a suitable diagram can be supported by linking `constraints` with respect to for example the thrust.

```
Discrete Input from Operator
     ----------------------------

     Class: Undefined!
     A_wing   aspect ratio of wing ...........................[-]:     9.10
     bypass_ratio
              bypass ratio of engine .........................[-]:        5
     b_f      max. width of major cross section of fuselage ..[m]:     3.30
     e        Oswald factor; picklist: .......................[-]:      0.8
     engine_config
              picklist: ...............................[string]:  ON_WING
     engine_type
              picklist: ...............................[string]: TURBOJET
     Flight_phase
              flight situations; pick list: ..................[-]:       CR
     h_f      max. height of major cross-section of fuselag ..[m]:     3.30
     l_f      length of fuselage .............................[m]:    27.88
     M        Mach number (V/a) ..............................[-]:      0.7
     Ne       number of engines ..............................[-]:        2
     sweep    sweep angle at quarter-chord line ........[degrees]:    15.00
     toverc_root
              thickness ratio at wing-root ...................[m]:    0.153
     toverc_tip
              thickness ratio at wing-tip ....................[m]:    0.122
     T_to_spec
              specific thrust at take-off (jet thrust/airflow [-]:    30.06
     Wto      max. take-off weight (MTOW) ....................[N]:   399200
     W_ratio  relative actual weight (W/Wto); picklist: ......[-]:        1

     Class: Constant
     gamma    ratio of specific heats of air [Ruijgrok 1994 ..[-]:      1.4
     pi       geometrical constant ...........................[-]:  3.14159
     p_0      atmospheric temperature at sea-level ISA [Ru ..[Pa]:   101325
     R        gas constant of air [Ruijgrok 1994, tabel  ..[Nm/K]:   287.05
     rho_0    atmospheric density at sea-level ISA [Ru ..[kg/m^3]:    1.225
     temp_0   atmospheric temperature at sea-level ISA [Rui ..[K]:   288.15

     Class: Correction factor
     r_f      zero-lift drag corr. factor for shape of fuse ..[-]:        1
     r_n      zero-lift drag corr. factor for engine instal ..[-]:      1.5
     r_t      zero-lift drag factor due to tail; picklist: ...[-]:     1.24
     r_thr    zero-lift drag corr. factor for thrust revers ..[-]:        1
     r_uc     zero-lift drag corr. factor for under-carriag ..[-]:        1
     r_w      zero-lift drag corr. factor for braced wings; ..[-]:        1
```

Figure 5.22. User input of the template.

The other specifications are dealt with in separate QUAESTOR sessions. Although QUAESTOR is able to construct a `template` which calculates more top-goal `parameters`, it is not possible to use one `relation` several times in one `template`. Since some basic `relations` will be used for several specifications in different situations, this limitation forces the user to construct several `templates`. Therefore for each specification a separate `template` is constructed

Recent developments of QUAESTOR allow for multiple use of `relations`, though, by applying an object-like syntax.

5.5.4 Performing calculations with the template

QUAESTOR is already calculating the `parameter` values while constructing the template. So, when the template has been completed, one value of the goal `parameter` is determined.

To perform sensitivity studies, the independent `parameter` values can be given other values. It is also possible to input ranges of values, which result in a range of values of the goal `parameter`. Together with the values of the other dependent `parameters` they can be saved in ascii-files and presented in graphs.

Figure 5.23 shows parts of the results of the example about the cruising condition. The goal `parameter ToverW` (the thrust-loading) is calculated as function of the independent `parameter WoverS` (the wing-loading) for one cruising Mach-number and two different cruising altitudes.

The graphical presentation of these `parameters` is shown in Figure 5.24. The results of only one template is presented; an external program is needed to show the results of all templates together.

When QUAESTOR detects a `constraint` to become unsatisfied, the user is prompted to reconstruct the `template`. The valid part of the `template` can still be used.

A loop in the `template` is handled by applying the Newton-Raphson and Simplex methods. The required initial values have already been defined in the control slots of each `parameter`. The accuracy of the approximated value has also been specified by the developer in a `parameter` control slot. Errors may occur when the approximation process diverges, for example because wrong initial values have been given.

```
Derived Discrete Values
-----------------------

Class: Undefined!
CD0S_f   zero-lift drag area due to fuselage ......[m^2]:   0.5704
CD0S_n   zero-lift drag area due to engine and nac [m^2]:   0.1491
toverc_mean
         mean thickness-chord ratio .................[-]:     0.14


Derived Multi-case Values
-------------------------

No.        a      alt      CD0   CD0S_t   CD0S_w        h kin_visc
       [m/s]     [ft]      [-]    [m^2]    [m^2]      [m]  [m^2/s]
-------------------------------------------------------------------
   1  303.15 30049.26   0.0151   0.4354   1.2439     9150 3.25E-05
   2  303.15 30049.26   0.0160   0.3928   1.0662     9150 3.25E-05
   :
   7  303.15 30049.26   0.0205   0.2862   0.6219     9150 3.25E-05
   8  296.61 34975.37   0.0156   0.4354   1.2439    10650 3.77E-05
   9  296.61 34975.37   0.0166   0.3928   1.0662    10650 3.77E-05
   :
  14  296.61 34975.37   0.0212   0.2862   0.6219    10650 3.77E-05


No.        q     Re_f  rho_atm     r_Re   S_wing     temp temp_rat
      [N/m^2]     [-] [kg/m^3]      [-]    [m^2]      [K]      [-]
-------------------------------------------------------------------
   1   10311 1.82E+08   0.4580     1.05   166.33   228.68     0.79
   2   10311 1.82E+08   0.4580     1.05   142.57   228.68     0.79
   :
   7   10311 1.82E+08   0.4580     1.05    83.17   228.68     0.79
   8    8201 1.54E+08   0.3805     1.08   166.33   218.93     0.76
   9    8201 1.54E+08   0.3805     1.08   142.57   218.93     0.76
   :
  14    8201 1.54E+08   0.3805     1.08    83.17   218.93     0.76


No.   ToverW  T_ratio        V
         [-]      [-]    [m/s]
-------------------------------
   1   0.269     0.28   212.20
   2   0.253     0.28   212.20
   :
   7   0.230     0.28   212.20
   8   0.287     0.23   207.63
   9   0.275     0.23   207.63
   :
  14   0.268     0.23   207.63
```

Figure 5.23. *QUAESTOR output from the template on the cruising condition.*
NB. The zero-lift drag area value due to the tail, `CD0S_t`, changes with the varying `WoverS`, because `CD0S_t` is related with the similar parameter for the contribution of the wing, `CD0S_w`; see Figure 5.21.

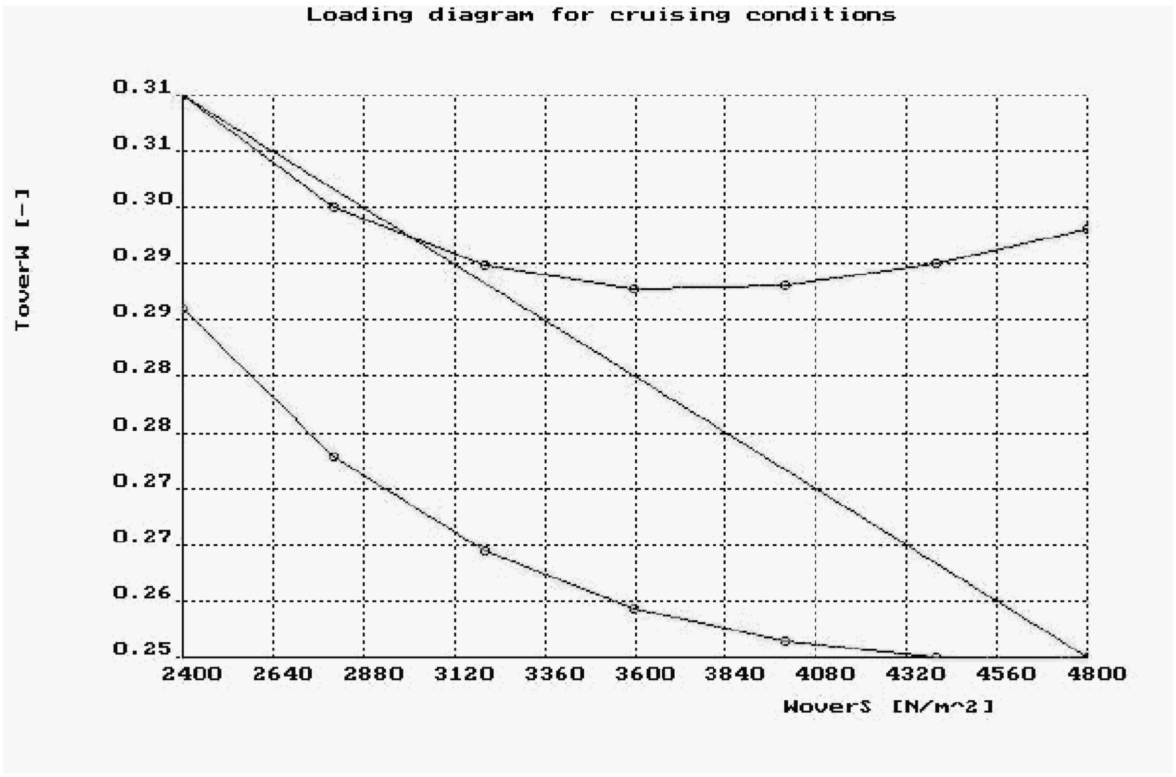


Figure 5.24. The graphical presentation of the calculation results for the cruising condition: the thrustloading `ToverW` versus the wingloading `WoverS` for the specified cruising speed and two different altitudes (upper line: h=10650m; lower line: h=9150m).
Note: the diagonal line in the diagram should be ignored; this is caused by the way QUAESTOR reads the data: per column.

5.5.5 Combining the results of several `templates`

For each specification a template is created. With these templates the influence of some important design `parameters` are studied, such as the maximum liftcoefficient in take-off and in landing configurations. When requested, the user copies the input values from the adapted case; that is: the wing data is originated from the British Aerospace RJ70, the other aircraft data from the Fokker 70. Some detailed engine data is copied from the Rolls-Royce BMW RB512 turbofan engine; this data was available due to a GARTEUR-project [Fransen, 1996]. An overview of the requested data is given in Table 5.7; the C_{Lmax} `parameter` for the landing and take-off situations as well as the wing- and thrust-loading are not included since they are subject to the sensitivity studies (and not copied from the case data):

Table 5.7. Requested (aircraft) case data and their sources for the specification calculations.

specification	Fokker 70	BAe RJ70 (wing)	RB 512 (engine)
take-off (AE)	-	A	λ^c
second segment climb (OEI)	-	A	-
ceiling (OEI)	d^a_{fus}, l_{fus}, W_{to}	A, Λ^b, t/c	λ^c, $T_{to}{}^I$, $T^d_{to,spec}$
landing distance	-	-	-
cruising speed and alt.	d^a_{fus}, l_{fus}, W_{to}	A, Λ^b, t/c	λ^c, $T_{to}{}^I$, $T^d_{to,spec}$

a. = fuselage diameter; b. = sweep; c. = by-pass ratio; d. = specific thrust at take-off
I. : T_{to} has only a minor effect on the determination of the drag-contribution of the propulsion.

The calculation results are put into one diagram, the loading diagram; see Figure 5.25. Together the different lines in this diagram frame the solution space, and an appropriate combination of wing-loading W/S and thrust-loading T/W can be chosen.

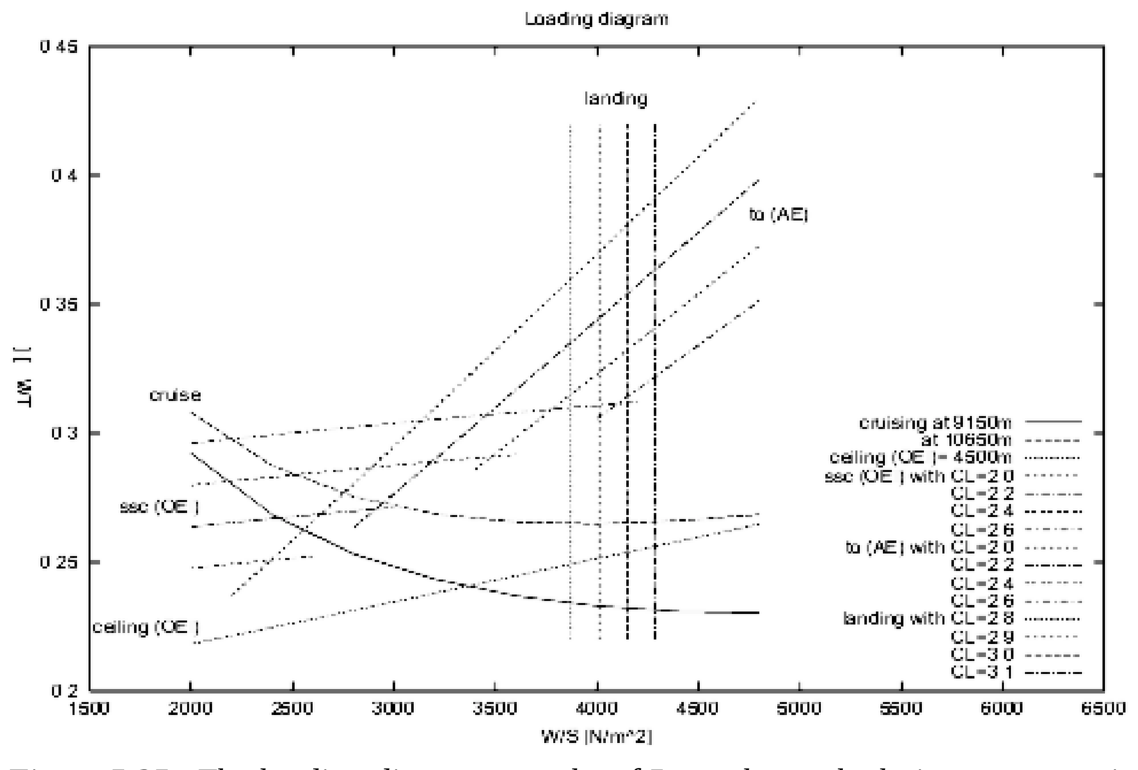


Figure 5.25. The loading diagram: results of 5 template calculations representing 5 design specifications.

In general the designer wants low values of the thrust-loading T/W, because this is translated into lower thrust engines which are less sophisticated and expensive, and/or increases the maximum weight limit. So a low thrust-loading gives the designer more design possibilities.

A high value of the wing-loading means a smaller wing is needed, which is cheaper and results in lower drag values. Higher wing-loadings also make the aircraft less sensitive to gust-loads, and therefore increases the comfort of the passengers. But because the wing usually contains (most of) the fuel, the wing should not be too small, i.e. the wing-loading value should not be too high.

With these considerations, the following values have been chosen:

- T/W = 0.29
- W/S = 3500 N/m^2
- C_{Lmax_TO} = 2.4
  A range of maximum liftcoefficients have been studied with respect to the take-off and landing distances. Knowing that a large value requires a complex high-lift system, a low value is favoured. Lower values result in longer distances, however. With the chosen value the take-off distance of the design will just be smaller than the specified maximum distance.
- C_{Lmax_land} = 2.8 (or 2.7 seems possible)
  The maximum liftcoefficient during landing will generally be higher than during take-off, because the high value of the induced drag is not a problem in the landing condition. The maximum liftcoefficient in the landing situation can usually be about several tenths higher, using the same high-lift system.

With the initial value of the maximum take-off weight of 400 kN (copied from the Fokker 70), this leads to:

- T_{to} = 58 kN per engine;
- S = 114 m^2.

When in the next design steps more details will be known about the weight distribution, the range can be calculated. In general, the initial weight value has to be modified to meet the range specification, and the loading diagram has to be generated again. Because the templates are saved within QUAESTOR, this iteration can easily be executed.

The loading diagram in Figure 5.25 can also be used to evaluate the specifications. A change in the cruising altitude, for example, seems to have a large effect on the design space: a small increase of the cruising altitude boosts the required thrust-loading.

The FAR-regulation about the minimum ceiling altitude with One Engine Inoperative does not seem to have a large impact on the design. The take-off specifications, however, are very restrictive.

5.5.6 Evaluation of the Functional module

The Functional Module is helpful, but it is clear that the user still needs domain knowledge to perform sensitivity studies. A skilled designer is needed to define the goal `parameters` which are helpful to evaluate some design `parameters`, to create the template in QUAESTOR, to choose the design `parameters` and to evaluate the results of the calculations.

With respect to the computer tool QUAESTOR, some particular difficulties are encountered which the user has to deal with. These are discussed in this section.

Choosing the right `relation`

It is not always clear which `relation` to choose from the presented list. One problem is that a `relation` can only once be used in a template; that is, a `relation` can be selected only once. The user should therefore have some thoughts about the `relations` he may need to build the template. And he should be aware of the implications of the alternative `relations`: which new `parameters` will they introduce?

QUAESTOR offers the possibility to reconsider a selected `relation`, though. Also when the template has been finished the user can un-select a `relation` and QUESTOR continues to support the reconstruction of the template.

In general, a proper overview of the (unfinished) `template` is very helpful when building it. Due to its limited graphical capabilities, the applied version of QUESTOR is not able to create some kind of graphical representation of the `template`. Such representations can become very complex when the `template`'s size increases, though, so that the overview problem would still exist.

Conditional validity of empirical relations

Empirical relations are based on simplified physics in combination with statistics. This makes them only valid under certain circumstances. In QUAESTOR this can be implemented in different ways:

- coupling `constraints` to a `relation`;
- not to automate it, i.e. let the designer decide whether or not the `relation` will be used.

When the circumstances are well-defined and have always to be met, the first option is implemented. When this is not the case, it is preferable to let the designer decide (latter option).

***Derivative* `relations`**

Because a `relation` can be used only once in a template, the knowledge base should contain the general and principal `relations` as well as some important derivatives. Only the main derivatives are included, since a large knowledge base is difficult to survey which makes it hard to select a proper `relation`. With QUAESTOR it is possible to perform the derivations 'on-line', but that requires unnecessary attention of the user.

A derivative `relation` is usually only valid for specific conditions. Therefore such a `relation` involves specific `parameters` and extra `relations`. For example, the derived equation for the lift-coefficient for minimum drag conditions:

$$C_{L(Dmin)} = \sqrt{\pi A e C_{D0}} \qquad \text{(Eq. 5.2)}$$

This `relation` is put directly into the knowledge base. It introduces a new `parameter`, $C_{L(Dmin)}$, and requires an extra `relation`, Equation 5.3:

$$C_L = C_{L(Dmin)} \qquad \text{(Eq. 5.3)}$$

Combined parameters are not decomposed

Often particular combinations of parameters are considered, such as the wing-loading *W/S* and the thrust-ratio T/T_{TO}. In fact, they are like any other parameter but lack an officially defined symbol. When these combined parameters are important, they are introduced separately; for example $WoverS = W/S$ and $Tratio = T/T_{TO}$.

To reduce redundancy, either the combined `parameter` or the separate `parameters` are used in a `relation`; that is, either a `relation` with the combined `parameter` is created or the same `relation` is created with the separate `parameters`. This may lead to the use of extra `relations` and extra `parameters` in the template, however.

It is not always clear which option to choose. When a combined `parameter` is important and can be calculated in different ways, it seems better to use this `parameter`. The dynamic pressure q is such a case, because it can be calculated in different ways: $q = \frac{1}{2}\rho v^2$ and $q = \frac{1}{2}\gamma p M^2$. Therefore the definition of the lift-coefficient will be formulated as: $L = qSc_L$.

***Automatic acceptance of a* `relation`**

The developer of the knowledge base should be very careful with the control attributes of the `relations` and the `parameters`. Some particular combinations of these control settings may lead to automatic acceptance of a `relation` in a specific situation. It is obvious that this is not desired when another `relation` is preferred, since multiple use of `relations` is not possible.

Multiple use of `relations`

New developments of QUAESTOR offer multiple use of `relations`. An index system has been developed for that purpose. However, these improvements have not been considered in this study.

'Reading' tables

In empirical relations often factors are introduced of which the values are pre-defined, depending on the circumstances. The values can be prescribed by the airworthiness regulations, or result from the designers experiences and statistics. In the references many tables with these factors, their values and their circumstances have been given.

In QUAESTOR a set of pre-defined values can be coupled to a `parameter`. To every pre-defined value a comment-line is added, so that there are no limitations to the description of the circumstances. When required, the designer can select one of the pre-defined values, or overrule it with a self-defined value.

The determination of the appropriate value can also be completely automated by implementing the table by a `relation`, using logical operators. In that case, however, the circumstances have to be explicitly described in algebraic notations.

'Reading' diagrams

Some numerical knowledge is put into diagrams, such as engine-data. In QUAESTOR this type of knowledge can be implemented using external data-files and interpolation functions in `relations`. The external file contains the data of the diagrams in a special format, called Telitab (see [Van Hees, 1997], written in ascii-code. In the `relation` a function-call can be assigned which contains the names of the external data-file and the interpolation `parameters`. The function-call will return an interpolated value, which is used in the `relation`. Several interpolation methods can be applied.

A diagram can of course also be implemented by a self-defined polynomial function.

Existing calculation modules

Also existing, independent calculation modules can be incorporated in a QUAESTOR template. They are considered as black boxes, from which the input and output `parameters` are defined. With a function-call in a `relation` the external module can be integrated in the `template`. The function-call must include the name of the module, and the input and output `parameters`.

5.6 Visualization of the aircraft concept with the Geometrical module

5.6.1 Introduction

The Geometrical Module creates a simplified, three-dimensional model of the external surface of the design. The model is visualized in order to help the designer understanding his creation, and it can be exported for the use of for more detailed calculations in the succeeding design steps. Because of these purposes the model has only a low level of detail.

The commercial software package Pro/ENGINEER© [Parametric, 1999] is used in this module. This package applies so-called features to define parts and combinations of parts (assemblies). Features are engineering entities such as protrusions, cuts and holes, which refer to the designer’s intentions. In addition constraints can be applied for more powerful parametric and flexible modelling. The feature and constraint definitions involve parameters which can be used in numerical expressions. Together these characteristics make Pro/ENGINEER a flexible tool for parametric modelling of the aircraft concept.

The Student Edition of Pro/ENGINEER, which has been applied for this project, does not allow automatic construction of the models. However, automatic suppression of elements of the model is possible. This technique is used for the definition of the aircraft concept.

For this reason standard aircraft components are used with pre-defined topologies[1]. The module asks for the values of some Structure parameters, and generates the model automatically.

The use of pre-defined parts corresponds well with the characteristics of the CBR module, i.e. generating new design concepts by combining existing configurations. A consequence is that when a new topology is loaded into the case-base, the developer should also update the pre-defined parts in the Geometrical Module.

1. The terms ’part’, ’component’, ’type’, ’configuration’ and ’topology’ may be confusing. In this thesis the following definitions have been used:
   - part: a physical entity of an aircraft; in Pro/ENGINEER a part is defined by a specific set of parameters;
   - component: in Pro/ENGINEER a component is a member of an assembly; this could be a part or a sub-assembly;
   - type: comparable with a part; for example: another type of wing is defined by another set of parameters;
   - configuration: the way parts have been located with respect to each other; a different configuration uses a different set of parameters;
   - topology: describes the set of definition parameters; different part-types are defined by different topologies, and different configurations are also defined by different topologies.

Because the parameter changes are unlimited, endless variations of shaping a topology are possible. These variations are only restricted when interferences of two parts occur, which can be checked by Pro/ENGINEER or by the user.

The organisation of the aircraft model is strongly hierarchical. The overall concept is modelled as an assembly of the main components: wing, fuselage, tail surfaces and engines. Topological constraints between these components define their relative positions.

The main components are also modelled as assemblies. These sub-assemblies consist of one or more parts, each defining another topology. For example the sub-assembly Engine consists of two parts, representing the turbofan engine and the turboprop engine. The appropriate part-type is selected by suppressing the other topologie(s).

Every part is built of one or more features. The features and the relation between them define the topology of the part. The parameters which are used by the features and the relations can be changed to shape the part.

5.6.2 The pre-defined components

Every type of component is defined as a separate part. The topologies are kept very simple to reduce the number of definition parameters.

Often there are more ways to define a component, applying different parameters. In general, those definitions are preferred which require the most abstract (Structure) parameters, such as non-dimensional and ratio parameters. The idea is that these parameters have more meaning to the designer than the low level parameters. When necessary, additional equations are defined to relate the higher level parameters with those of the lower level.

The part Straight Tapered Lifting Surface, for example, is defined by the Pro/ENGINEER feature ’protrusion’, which requires the definitions of a set of cross-sections, their relative positions and the distances between the sections. For a ’straight tapered’ surface only two cross-sections are defined; a ’cranked’ surface requires three sections. Next figure shows the part

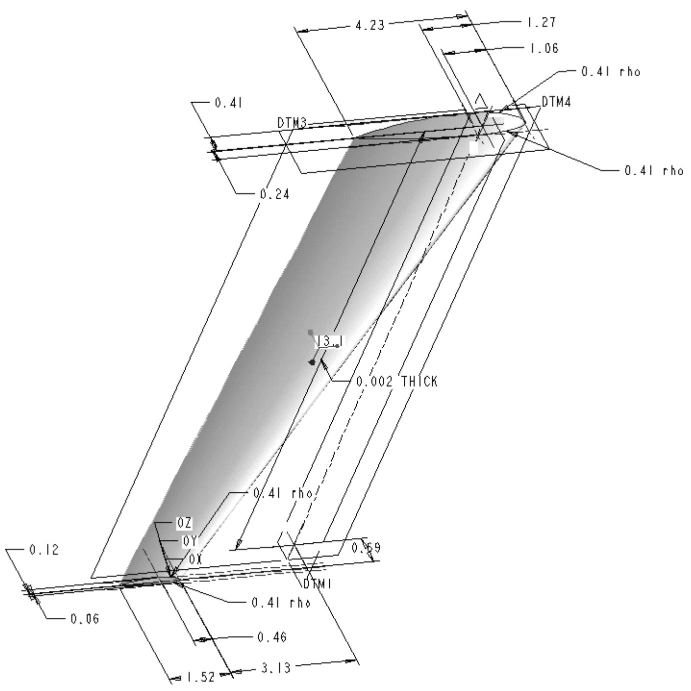


Figure 5.26. A view of the part 'Straight Tapered Lifting Surface' in Pro/ENGINEER.

Because the parameters required by the feature 'protrusion'[1] are very low-level, extra equations are attached to the part. Through these relations the shape of the lifting surface can be defined by the aspect ratio A, the surface area S, the taper ratio λ, and the sweep of the 1/4-chord line $\Lambda_{1/4}$. The surface area S can be determined by other, more abstract parameters, such as the volume coefficient of the tail surfaces; these relations are defined at the assembly level, a higher level; see next section.

Some parameters have been given pre-defined values. These parameters are required to define the geometry, but are of no importance to the designer at this design phase. For example the definition of the cross-sections: symmetric NACA airfoils have been used. In Pro/ENGINEER this is implemented by an internal program file which is attached to the model.

To be able to define its location and attitude, every part contains at least three virtual planes which are used as references. When possible these planes are defined to have a physical meaning to the designer. For example the longitudinal plane through the Mean Aerodynamic Chord (MAC) of the lifting surface, and the perpendicular plane through the 1/4-chord point of the MAC. The use of these planes is discussed in next section about the topological constraints.

1. The feature 'protrusion' can model more than just the physical protrusion. Cross-sections may change and the protrusion-traject does not have to be a straight line. The 'ruled surface' of AutoCAD is a comparable feature.

5.6.3 Topological constraints

Topological constraints determine the type of component to be modelled, and the way a component is positioned with respect to another component; i.e. the constraints define the *type* and the *configuration* of the aircraft components, respectively.

As is mentioned in the Introduction, the Geometric Module contains all pre-defined types of components. To present the appropriate components in Pro/ENGINEER the non-selected types or parts are suppressed. This is straightforwardly implemented in the internal program file of the assemblies by <if>..<then>.. rules and a 'suppress' command.

The configuration constraints are defined with the aid of reference planes. Every aircraft component has at least three independent reference planes, as has been described in the previous section. With the determination of the distances between these planes, their location and attitude with respect to each other is fixed. More reference planes have been introduced to be used by other components and to allow alternative reference definitions which are needed for other topologies.

Some of the reference planes correspond to planes with a physical meaning. For example the vertical spanwise plane through the 1/4-chord point of the Mean Aerodynamic Chord (MAC) of the wing. This reference plane is used to define the longitudinal position of the wing with respect to the centre of the fuselage.

Another vertical reference plane of the wing coincides with the root. This plane coexists with the vertical similarity plane of the fuselage, to define the spanwise location of the wing.

The vertical location of the wing is defined by the discrete parameter 'wing_config'. With this parameter it is indicated whether the aircraft has a low-, mid- or high-wing configuration. Depending on this parameter the wing is modelled below the fuselage, through the fuselage or on top of the fuselage, respectively.

Figure 5.27 shows the topological constraints between the wing and the fuselage.

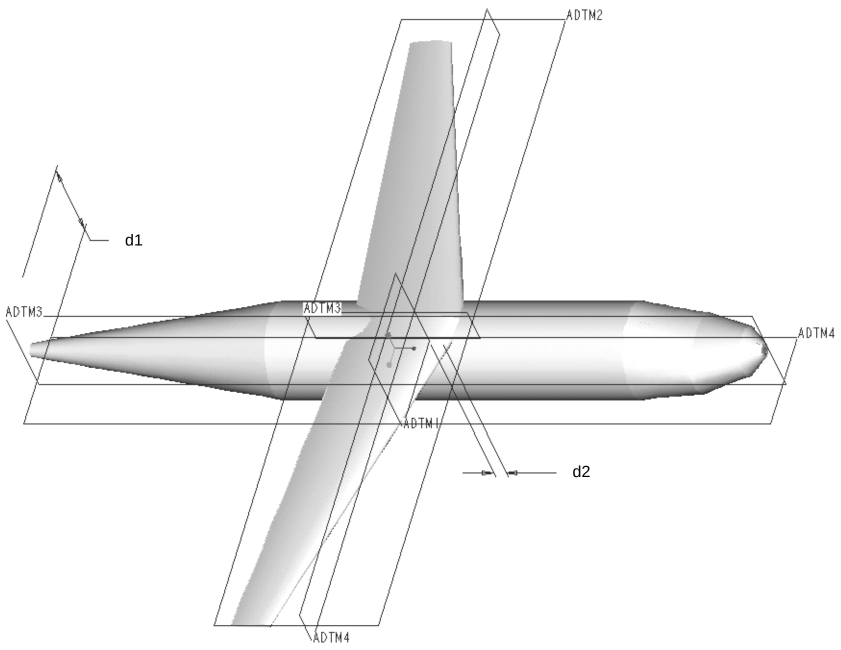


Figure 5.27. The topological constraints between a wing and a fuselage.
- offset=0: between datum planes ADTM3 of the wing and ADTM3 of the fuselage;
- offset=d1: between datum planes ADTM2 of the wing and ADTM4 of the fuselage;
- offset=d2: between datum planes ADTM4 of the wing and ADTM1 of the fuselage.
Parameters d1 and d2 are driven by input parameters describing the wing-configuration (high-wing) and the longitudinal wing location, respectively. Datum plane ADTM4 of the wing is located at the 1/4-chord point of the Mean Aerodynamic Chord (MAC).

The implementation of the topology constraints are defined at the top-assembly (complete aircraft) and the sub-assembly (components) levels. Many involved relations are implemented in the internal program files, which are attached to the models.

Within Pro/Engineer the parameter values can only be forwarded from higher to lower levels, i.e. from super-assemblies to sub-assemblies to parts. However, some relations require parameters from the lower part level. Because of this problem many relations are defined at the top-level of the aircraft model, which makes the overview of the internal program rather complex and prevent an decentralized organisation of the model.

When the aircraft model has been defined, it is possible to smooth and polish it within Pro/ENGINEER. This may be interesting for calculations further in the design process. These case-specific modelling techniques have not been implemented in the AIDA project, however, because this complicates the automatic generation process extremely, and it is beyond the purpose of the project.

5.6.4 The input

The Geometrical Module is started by loading the general model of the aircraft into PRO/ENGINEER. The internal program which is attached to the model will be run and asks for the values of the input parameters. These parameters are Structural parameters, and most of their values are copied from the case-data which have been selected by the CBR-module, i.e. from the ’best-matching’ and the ’adaptation’ cases. Other parameters have been examined in the Functional Module, so their values originate from the sensitivity studies. The values can be collected in a simple input-file to ease the input process. For a smooth operation of the tool the input-file will be generated automatically to increase the integration of the modules; that is a recommendation for future work.

With this set-up the Geometrical Module does not receive redundant and possible conflicting data. The problem of redundancy is forwarded to the compilation of the data-file.

Most of the parameters come directly from the ’best-matching’ and the ’adaptation’ cases. Generally it is clear from which case to copy the value, depending on the components that have been mixed. For some conditions this may be difficult, however, and the knowledge of an expert is required.

Some parameters are omitted when they have no meaning for the current configuration. For example the location of the wing crank is useless for straight tapered wings.

5.6.5 The output

The Figure 5.28 shows a 3-dimensional isometric view of the aircraft design; Figure 5.29 shows the three views on the model.

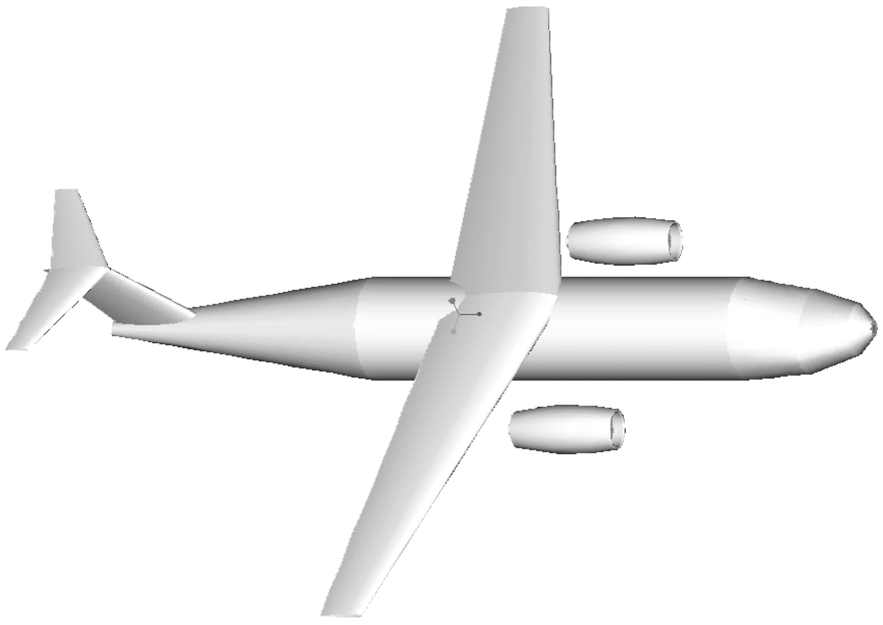


Figure 5.28. A shaded, 3-dimensional view of the design.

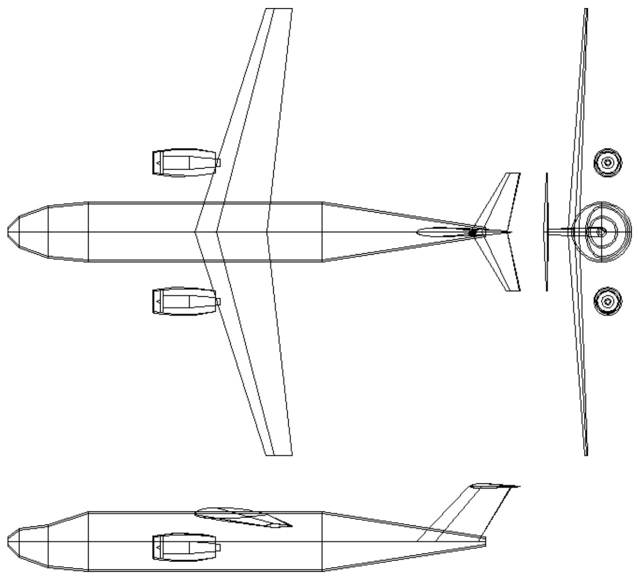


Figure 5.29. The 3 views of the design.

Pro/ENGINEER is able to export the model in different formats, such as IGES and STEP. This may be required by the succeeding design tools. Due to the other formats, though, the parametric set-up of the model will be lost.

5.6.6 Evaluation

The visualization of the model helps to assess the design concept. The set of definition parameters offers sufficient flexibility to model a wide range of configurations. The model retains some imperfections, though, such as the overlappings between the wing and tail surfaces and the fuselage, and the lack of engine pylons and undercarriages. It is possible to repair these imperfections, but that will increase the complexity.

The parameterisation of the model offers good flexibility to change data values. This is useful when doubts arise about some aspects of the concept.

For example the sizes of the tail surfaces in this case study: they seem to be too small in Figure 5.28. Their sizes are directed by the volume coefficients:

$$\text{volume coefficient} = \frac{\text{distance wing to tail} \cdot \text{tail surface area}}{\text{reference wing length} \cdot \text{wing area}} \qquad \text{(Eq. 5.4)}$$

In this case study, the volume coefficient values are copied from the ’best-matching’ case, the Fokker 70, whereas the wing geometry is copied from the ’adaptation’ case, the BAe RJ70. The values of the Fokker 70 are about 40% smaller then the values of the BAe RJ70, resulting in smaller tail surfaces for the Fokker 70.

The question arises whether this has been the right choice. An expert could argue that the volume coefficients are dependent on the engine configuration: with one engine inoperative the yawing moment will be smaller for the Fokker 70 with fuselage-mounted engines, than for the BAe RJ70 with its wing-mounted engines. With this expert knowledge the user may want to reconsider is initial choice, and use the volume coefficients of the BAe RJ70.

This dilemma is independent of the applied modelling techniques, though; it is about the composition of the data-file of an adapted case, i.e. a case that is made up of two cases (see Section 5.4). Due to the flexibility of the parameterised modelling technique and the choice of the definition parameters, the user can easily revise his choice and evaluate the results of another choice: copy the coefficient values of the BAe RJ70. The resulting design is shown in Figure 5.30 and Figure 5.31; according to the authors impression this design is more natural.

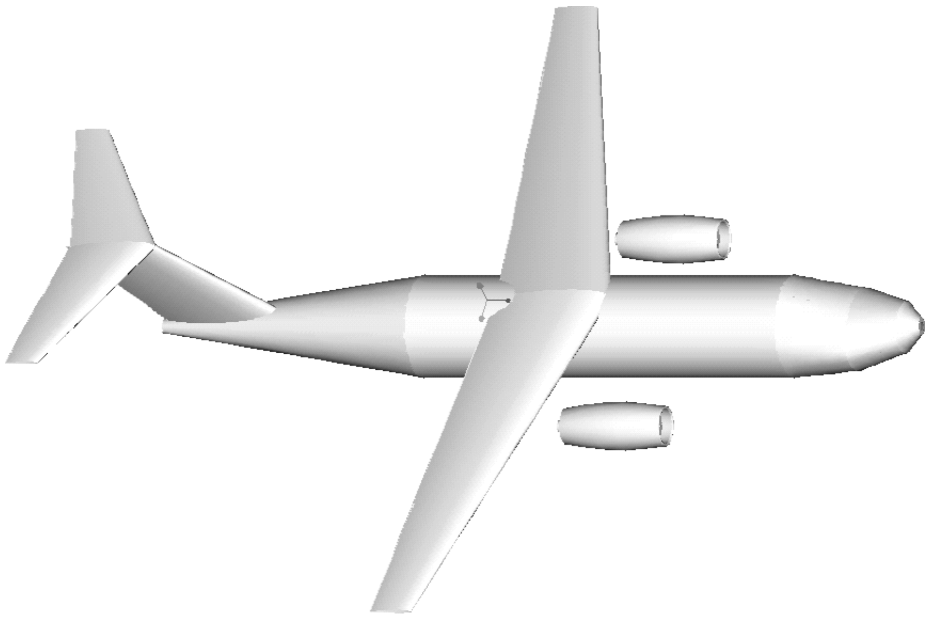


Figure 5.30. Shaded view of the design with larger tail surfaces.

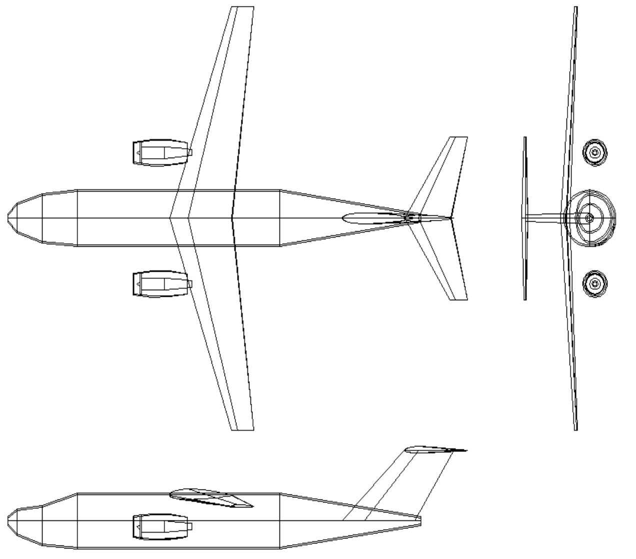


Figure 5.31. The 3 views of the design with larger tail surfaces.

These specific modifications have no effect on the previous calculations in the Functional Module. When a reconsideration of the calculations is required, though, the user can easily return to that module.

There are several aspects of the model which may need some upgrading, such as the overlapping of components. The current implemented model clearly demonstrates the possibilities and impossibilities of the applied techniques, however.

An aspect of Pro/ENGINEER that has not been exploited yet is the use of physics related functions, such as the calculation of volumes and wetted surface areas. These quantities are useful in more detailed calculations in the next design steps. A feedback loop with the Functional Module could be possible.

Pro/ENGINEER is also able to determine masses and centres of gravity of the components, but to get realistic and useful values the detail level should be increased extremely.

5.7 Evaluation of the case study

It is difficult to evaluate the result of the case study, i.e. the design concept. With the Functional Module, the influence of some sizing parameters can be studied, but there is less relation between the topological parameters of the aircraft and the aircraft performances. At this level of detail the effect of another location of the engines on the performances, for example, is hardly perceptible. Only when the design is elaborated into more detail, these effects become clear. This could be performed with tools that are suitable for the next design phase, such as ADAS [Bil, 1989] or other tools mentioned in Chapter 2. Since this is beyond the scope of this study, such evaluation has not been worked out.

A kind of evaluation can be carried out by comparing the overall result of the case study with the results of the Garteur-project mentioned in the Introduction, which started from identical design specifications; see also [Fransen, 1996]. This project led to the same configuration, which at least gives confidence in the validity of the support that the AIDA system provides.

With respect to the design process, the AIDA modules were able to provide enough support. The adaptation with the aid of the CBR module, however, is very subjective. The presented concepts by the CBR module are helpful, but still quite some domain knowledge is required to be able to use them well. Domain knowledge is also required in the other design phases, but this seems to be less distinct. Only the use of the Geometrical module hardly needs attention of the designer, although the character of the geometrical model may be too primitive.

6 Conclusions

In this final chapter the achievements of the AIDA project are reconsidered. The first section, Section 6.1, describes how far the implementation has reached the goals which have been set in the first chapter. The discussion concentrates on the usefulness of the separate modules. In Section 6.2 the appropriateness of the applied techniques is discussed more generally, based on the lessons learned from the AIDA project.

6.1 Evaluation of the AIDA system and its modules

6.1.1 The AIDA set-up

The AIDA set-up bears a strong resemblance with the general design cycle, and contains the following steps:

1. selecting an appropriate starting point from a case-base,
2. adapting it by combining several cases,
3. modelling (functional and geometric) and visualizing the concept;
4. modifying some sizing parameters via sensitivity studies.

The individual steps have been implemented in separate modules. Each module applies a different AI-technique, such as CBR (Case-Based Reasoning) and RBR (Rule-Based Reasoning), or a Geometric Modelling technique. Little attention has as yet been paid to the integration of these techniques and the communication between the modules. Indeed, the main concern of this study was to examine the usefulness of different AI-techniques in the conceptual design process. During the course of the project the study focused on relatively conventional aircraft configurations.

The case study has proven that the applied AI- and Geometric Modelling techniques are useful to support the conceptual design process. Despite the low level of communication between the modules, this set-up allows to relatively easily perform several iteration and adaptation steps without much overhead work. However, much effort is required in the preliminary phase, when the domain knowledge has to be stored into the case-base and rule-base using the appropriate data structure. It is also concluded that expert knowledge is still required when using the AIDA system. It is evident that it is impossible to put all this knowledge into the different modules.

It appears that apparently minor practical issues, specific for the tools, can have a significant impact on the overall performance of the system. For example the effect of missing case-data can decrease the similarity score considerably with the applied CBR tool.

On the other hand, such minor topics force the designer to think about almost every aspect of the reasoning involved in the case selection process. Developing such modules is therefore very helpful in understanding the process.

6.1.2 The CBR-module

The goal of the CBR-module is to support the generation of an initial aircraft concept, from which one or more feasible concepts are deduced. Chapter 5 (Section 5.3 and Section 5.4) shows what the supporting role includes: presenting existing aircraft in decreasing order of similarity with the design specifications. The designer defines the specifications, runs the CBR selection process, and selects one (or more) aircraft from the presented list. The case study in Chapter 5 reveals to what extend the module is able to support the designer: the module indeed finds appropriate cases, but the designer still needs domain knowledge to define the specifications and judge the similarity results properly. Especially when the CBR-module is used the second time for the adaptation of the initial aircraft concept, a thorough knowledge about the domain is necessary.

In addition the designer should have knowledge about the working of the CBR-module. Some aspects of this particular module (based on EADOCS of [Netten, 1997] need special attention and/or need to be improved:

- the determination of the global similarity number (needs special attention)
  The global similarity number is the result of a weighted summation of the local similarity numbers, i.e the similarities of the individual parameters. The designer can influence the similarity results by modifying the weighting functions and by changing the local similarity measurements. The weighting function is controlled by the priority factors and the object-oriented data structure. The local similarity is largely influenced by the parameter domains. Hence, to interpret the global similarity number, the designer should be well aware of these aspects.
- the organization of the data within each case (needs special attention)
  The used CBR-tool applies an object-oriented structure to the case-data. This structure plays an important role in the determination of the similarity of the cases with the specifications. Therefore a good understanding of the data structure is required to translate the design specifications properly into the target-file and to interpret the presented order of similar cases.
- the definition of the parameter domains (needs improvement)
  The matching of a single parameter is determined with the aid of parameter domains. When the values of both the target parameter and the case parameter fit within the same parameter domain, they score 100% similarity; otherwise the local similarity is 0%. For continuous parameters it does not matter if their values

are located at the centre of the domain or near the boundaries. Due to this method of similarity measurement, the parameter domains have to be defined with great care.

- the measurement of the similarity of individual parameters (needs improvement)
  As is discussed in the previous item, the current practice of similarity measurement is based on fitting values on pre-defined parameter domains, and result in a 100% or a 0% similarity score. This is a feasible method for integer and string parameters, and when parameter values can be categorized into distinct classes. For most of the continuous parameters, however, this is not a desirable method. It would be more realistic to express the similarity numbers between 0% and 100%, in some way proportional to the distance between target and the case value. However, this would require serious modifications in the organisation of the indexing network, which is based on the discrete parameter domains. This should be an important criterion when considering other implementation tools.
- identical conditions of the aircraft performance parameters (needs special attention)
  The similarity of existing aircraft is deduced by comparing their performance parameter values with the specified values. For a proper comparison the conditions of the performance parameters should be equal. For example flight-ranges of aircraft can only be compared when their flight conditions are known: cruising at maximum speed or for maximum range, etc.. These conditions should be well-defined for each parameter, or should be expressed by additional parameters and added to the similarity measurements, i.e. added to the same target object within the target file. This critical issue is inherent to the comparing operation.
- the matching of combined parameters (needs improvement)
  In Section 5.4.6 it is discussed that sometimes similarities are best determined by considering combinations of parameters, such as Payload and Range. The matching of a combination of Payload and Range results in more realistic similarities than the matching of each parameter individually, as is explained with Figure 5.14. The implementation of this type of matching is not possible with the applied CBR-tool, however.
- the completeness of the case-data (needs special attention)
  The applied CBR-tool returns a 0% similarity score of a single parameter when the case does not contain information about that parameter. Hence the developer of the case-base should put effort in generating the case-data as complete as possible. When the values have to be estimated one should be well aware of the defined parameter domains.

6.1.3 The Functional module

Within the AIDA project the intention of the Functional module is to offer a flexible tool for performing sensitivity studies. These studies are used to modify some primary (continuous) parameters of the aircraft concept, such as the wing area and aspect ratio. The initial values of these parameters have been generated with the aid of the CBR module and can be improved to meet the design specifications more closely. Flexibility is needed to deal with the large variety of parameters and to be able to use various (fundamental or empirical) relations, subject to the designer's preferences.

From the case study in Chapter 5 it can be concluded that the RBR tool QUAESTOR ([van Hees, 1997]) offers this numerical flexibility. Particularly the feature that a QUAESTOR relation can be used to calculate a parameter in the left-hand side of an equation as well as in the right-hand side is very useful.

A critical issue is the overview of the relational network and the relations in the knowledge base. Especially when the network grows larger it becomes difficult to keep a good overview, which is necessary to select the appropriate relations. This is a general problem of RBR systems, however.

The applied version of QUAESTOR has only minor graphical features, for example to show the results of various calculations in one graph. Therefore additional programs have been used. Newer versions of QUAESTOR have much improved graphical capabilities.

As is the case with the CBR-module, the designer has to have knowledge of the working of QUAESTOR and of the relations within the knowledge base for proper usage of the module.

6.1.4 The Geometrical module

The Geometrical module is able to model and visualize the aircraft concept automatically with a small set of lay-out and primary parameters. With the 3-dimensional shaded view and the 2-dimensional side-view capabilities of the Pro/ Engineer software the designer gets a good impression of the concept.

To automate the modelling process, strict definitions of the aircraft components and their lay-out have been applied. A disadvantage is that this reduces the flexibility of the module with respect to new component and lay-out definitions. On the other hand, this flexibility is not necessary since all components and lay-outs come from the case-base of the CBR module; hence they are known in advance.

Only new combinations of existing lay-outs may lead to problems with regard to the definitions. However, with intelligent definitions such as suitable assembly conditions (i.e. reference planes) those problems could be minimized.

While the developer of the Geometrical module has to know the package Pro/ Engineer well, the designer hardly pays attention to the module. The single concern of the designer is to assign appropriate values for the pre-defined set of parameters. Further interaction is limited to changing views and creating an export file for other (geometry oriented) programs.

6.2 AI in the design process

6.2.1 The CBR techniques

The term Case-based reasoning is very comprehensive. In Chapter 3 it is discussed that the complete CBR methodology resembles the design cycle and may cover more than just one AI technique. However, when referring to CBR as one single AI technique, the tasks of retrieving, reusing and retaining cases are implied. Other techniques such as RBR can be used to aid these tasks. How different AI techniques can cooperate together is discussed at the end of this chapter in Section 6.3.

The retrieving task is one of the most critical issues of the CBR technique. The system developer does not only meet implementation problems which are specific for the applied tool, such as mentioned in Section 6.1.2. He also has to cope with fundamental issues such as the determination of the similarity of cases, on which the selection of the best case is based. This issue deals with the problem of capturing the designer's judgment strategy or reasoning. Or in other words: how does the designer compare apples with pears? Is it possible to list all (design) criteria, quantify them, give them the appropriate weights and accumulate them for a proper judgment? Is an objective judgment achievable at all? And if that is possible, will those weights change when different combinations of quantified criteria are considered?

The point is that to select a 'best case' from a case-base is already a major problem for designers, leading to different 'best cases' for different designers; hence this is probably extremely difficult to formalize for use in some reasoning technique. In this respect the development of a CBR-based support tool can be considered as a tool to help analyse and systemize the design process and reveal (some) domain knowledge.

An identical discussion can be held for the selection of a case for the adaptation phase. In addition another fundamental issue arises: adapting a case disconnects the (complex) implicit relations within the case between the (function and structure) parameters. Without a good understanding of the domain knowledge, the combination of different cases may result in unrealistic designs which form inadequate starting points for the design process; Figure 6.1 shows an example for the case study. Consequently, adaptation may make a case worthless as a starting point.

Although very complicated, the adaptation phase is crucial for the capability of CBR to support *innovative* design in addition to *routine* design; see the discussion in Section 3.2.9. By combining the lay-out of different cases an original lay-out can be created. This has been shown in the case study in Chapter 5. Design which comprises new components and new parameters, classified as *creative* design, does not seem to be suitable for support by CBR.

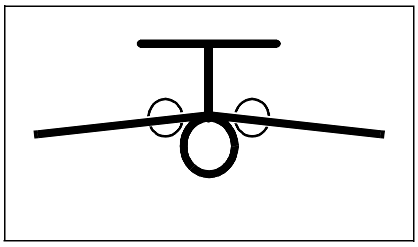


Figure 6.1. Example of an unrealistic design due to bad adaptation: the combination of high wings and fuselage-mounted turbofan engines.

Another major issue in CBR is the representation of the cases. This matter has large impact on the applications and on the implementation of the CBR tool. A representation by a minimum set of parameters, for example, simplifies the reasoning on the case data. Such a concise case representation would minimize the case-base and reduce the indexing problem. As a consequence this data should have a high abstraction level. This minimum case representation is favourable from the adaptation point of view, as it facilitates the mixing of data from different cases.

With the tools applied in this thesis, such a minimum representation results in a limited flexibility, because only a limited number of parameters could be used for the matching criteria. Therefore redundant parameters have been included in the case representation, which were derived from the abstract parameters. This increase in flexibility has been achieved at the costs of a more complicated indexing structure. To have the best of both sides, a CBR tool should be used which is able of transforming the (abstract) case parameters into the desired matching parameters before case matching is performed. In general this means that such capabilities make it possible to measure the similarity between cases from any point of view.

On the other end of the spectrum one could choose for a very comprehensive case representation, containing as much information about the case as possible. For example the relation networks generated by the Functional Module could be included. This means moving into the category of *derivational* CBR which comprises the reuse of solution methods instead of solution data. Although this extra case information could be very useful, it could also complicate the issues of case indexing and matching. The mixing of data of two cases becomes complex too, hence this type of representation is not recommended for the adaptation process.

6.2.2 The RBR techniques

RBR techniques have been implemented in many different forms. The applied RBR tool QUAESTOR [van Hees, 1999] is numerical oriented: it is very powerful in the handling of algebraic relations. This is appropriate for supporting sensitivity studies, which is the goal of the Functional module within the AIDA project. Because the algebraic expressions are well documented in the aircraft design handbooks such as [Roskam, 1987-90] and [Torenbeek, 1982], the knowledge acquisition is not difficult.

The main issue is when to use which relation. Basically, QUAESTOR suggests a list of relations based on syntax only, and the designer selects one of them. This selection is guided by many arguments: what is the global purpose of the relational network, i.e. what are the subjects of the sensitivity studies, and what are the consequences of the selection, i.e. which new parameters will be introduced and how difficult are they to be calculated.

So, the supporting role of RBR could be improved by aiding the selection process. It is difficult to formalise these selection criteria, however. One way of improvement would be to offer the designer a better overview of the (unfinished) relational network and the relations in the knowledge base, in order to make it easier to foresee the consequences. When the network becomes too big this overview becomes a major problem. It is still a challenge to define a suitable graphical format of representation for this.

Also for non-algebraic issues RBR techniques can be helpful. For instance some of the expert knowledge which is applied when using the CBR module and the Geometrical module seems suitable to be formalised into if-then-rules. In the case study in Chapter 5 some illustrations have already been given. See for example Table 5.7, which suggests to modify specific design parameters when certain performances are not met. Such explicit suggestions could easily be formalised. One could argue however the validity of those simple relations: here the knowledge acquisitions becomes a major issue.

Another direction for improvement is the link between the Functional and the Geometrical modules. When both modules can easily communicate with each other, it is possible to better organize the tasks of the modules; to shift algebraic relations which express geometrical relations from the Geometrical module to the Functional module. On one hand this would increase the flexibility and decrease the number of complicated relations which are implemented in the Geometrical module. On the other hand this would enlarge the knowledge base of the Functional module and its relational network, and therefore reduce the required overview. Whether the advantages would predominate the disadvantages largely depends on implementation details.

When the communication between both modules is improved, the Geometrical module could provide the values of typical geometric parameters, such as volumes and surface areas, which could be used by the Functional module. Such parameters can easily be calculated with standard functions.

6.2.3 The Geometrical modelling techniques

In the set-up of the AIDA system the design is not modelled on-line, but is built from pre-defined components. This strategy does not limit the practicability of the AIDA system more than the CBR module has already done; see Section 6.1.2.

Considering these limitations the applied parameter modelling techniques offer enough flexibility. In Pro/Engineer all three types of geometry parameters can be used as variables: the position and orientation parameters, the geometrical parameters and the topological parameters; see Section 3.3.3. However, the relations between these parameters have to be organized in a strict hierarchical structure. Although this leads to a simple CSP (Constraint Solving Problem), it also limits the flexibility of the geometric model: only unidirectional (geometric) relations are possible, for example. Hence the geometrical model should be pre-defined with great care.

Several recommendations can be given. With the tools that are currently implemented in the AIDA system, progress can be found in a closer cooperation with the Functional Module. Both the Geometrical Module and the Functional Module apply different types of CSP techniques, each with its deficits and benefits. When the benefits of both techniques could be optimal combined, the flexibility of the AIDA system should be increased. Using the advanced modules of the Pro/Engineer software package may also increase the possibilities for interfacing.

For further research it is recommended to select or develop tools that integrate the functionality of both the Functional as well as the Geometrical Module. This requires an integration of several CSP techniques. It is expected that Feature-Based modelling, as a further development from parameterised modelling, can be very powerful.

6.3 General recommendations

As mentioned before, improvements are achievable by integrating the different modules and reasoning techniques. For the RBR and the Geometric modelling techniques such integration makes sense when both techniques are considered as types of CSP (Constraint Solving Problem) techniques. Combining these modules should benefit from the flexibility of the RBR Module and the dedicated geometric reasoning of the Geometric Module.

Other integration aspects include the usage of RBR techniques in the CBR module, for situations when the expert knowledge seems to be very explicit and suitable for formalisation, and an integrated architecture of the complete system.

Another issue of great impact is the issue of case representation. Case representation may vary between a minimum set of abstract parameters to a comprehensive set that also includes methods. To reduce the complexity of the case matching and adaptation process as much as possible, minimum case representation would be preferred. For a maximum flexibility with respect to the usage of the cases, however, the cases should contain much more data. To combine both characteristics, a CBR module should be developed which is suitable of expanding the case data before case matching is performed. Such an approach requires a complete different indexing structure than applied in the current CBR module.

The recommendations mentioned above mainly concern the implementation of the techniques. More fundamental is the issue of case adaptation. Although adaptation has increased the applicability of CBR from routine design to configuration and perhaps to innovative design, this study has revealed many critical issues of adaptation. In this thesis one adaptation method has been elaborated upon. It turned out to be rather complicated and requires a thorough knowledge of the domain as well as of the implementation of the CBR module. A thorough study is recommended to improve this adaptation method or to examine alternative methods.

References

Aamodt, A., Plaza, E., *Case-based reasoning: foundational issues, methodological variations, and system approaches*, in: *AICOM*, Vol. 7, nr. 1, March 1994, pp. 39-59.

Anemaat, W.A., Schueler, K.L., Koffort, C.T., *General aviation airplane design tools for PC's*, SAE 971473, presented at: *General, Corporate and Regional Aviation Meeting and Exposition*, Wichita, Kansas, USA, April 1997.

Bil, C., "*Development and application of a computer based system for conceptual aircraft design*", PhD thesis, Delft University of Technology, The Netherlands, 1988.

Bil, C., *ADAS: a design system for aircraft configuration development*, AIAA 89-2131, 1989.

Bleijenbergh, J., *Chaince; a multi-user framework for aircraft design*, MSc thesis, Delft University of Technology, The Netherlands, 1997.

Blessing, L., *Engineering design and artificial intelligence: a promising marriage?*, in: Cross, N., Dorst, K., Roozenburg, N. (eds), *Research in design thinking; proceedings of a workshop meeting, held at the Faculty of Industrial Design Engineering*, Delft University of Technology, The Netherlands, May 1991.

Britton, C.L., Jenkinson, W.W., *A computer aided design for airplane configuration*, in: Murthy, T.K.S., Fielding, J.P. (eds), *Computer applications in aircraft design and operation*, Computational Mechanics Publications, Springer-Verlag, 1987.

Brown, D.C., Chandrasekaran, B., *Design problem solving; knowledge structures and control strategies*, Pitman Publishing, UK, 1989.

Buckley, M.J., Fertig, K.W., Smith, D.E., *Design Sheet: an environment for facilating flexible trade studies during conceptual design*, AIAA 92-1191, 1992.

Coyne, R.D., Rosenman, M.A., Radford, A.D., Balachandran, M., Gero, J.S., "*Knowledge-based design systems*", Addison-Wesley Publishing Company, 1990.

Denisov, V.E., *Application of methods and tools for computer-aided design in investigation of prospects for civil aircraft progress*, ICAS-90-2.1.4, in: *17th Congress of the International Council of the Aeronautical Sciences*, Stockholm, Sweden, September 1990.

Dohmen, M., *A survey of constraint satisfaction techniques for geometric modeling*, in: *Compu-ters & Graphics*, Vol. 19, No. 6, 1995. pp. 831-845.

Domeshek, E.A., Herndon, M.F., Bennett, A.W., Kolodner, J.L., *A case-based design aid for conceptual design of aircraft subsystems*, from: *Proceedings of the 10th IEEE Conference on AI for Applications (CAIA-94)*, San Antonio, Texas, USA, March, 1994, pp. 63-69.

Dym, C.L., *Engineering design; a synthesis of views*, Cambridge University Press, 1994.

Elias, A.L., *Computer-aided engineering: the AI connection*, in: *Astronautics & Aeronautics*, July/August 1983, pp. 48-54.

Elias, A.L., *Knowledge engineering of the aircraft design process*, in: Kowalik, J.S. (ed.), *Knowledge Based Problem Solving*, Prentice-Hall, 1986, pp. 213-226.

ESDU, *ESDU: Engineering Sciences Data Unit*, several volumes.

Fransen, S.H.J.A., *Conceptual design of a 70 passenger regional airliner propelled by fuel-efficient turbofan engines*, memorandum M-725 of the Faculty of Aerospace Engineering, Delft University of Technology, The Netherlands, March 1996.

Fransen, S.H.J.A., Torenbeek, E., *Derivation and results of an ADAS program for the Dutch Green Aircraft Pilot-Study*, memorandum M-735 of the Faculty of Aerospace Engineering, Delft University of Technology, The Netherlands, October 1996.

Gero, J.S., Tham, K.W., Lee, H.S., *Behaviour: a link between function and structure in design*, in: Brown, D.C., Waldron, M., Yoshikawa, H. (eds), *Intelligent Computer Aided Design*, Elsevier Science Publishers B.V., The Netherlands, 1992, pg. 193-221.

Gevarter, W.B., *The nature and evaluation of commercial expert system building tools*, in: *Computer*, Vol. 20, No. 5, May 1987, pg. 24-41.

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

Gonda, M., Fertig, K.W., Teter, R.J., *Aerospace conceptual vehicle design using an intelligent design and analysis environment: Design Sheet*, AIAA 92-4222, 1992.

Haberland, C., Thorbeck, J., Fenske, W., *A computer augmented procedure for commercial aircraft preliminary design and optimization*, ICAS-84-4.8.1, 1984.

Hees, M. Th. van, *QUAESTOR expert governed parametric model assembling*, PhD thesis, MARIN/ Delft University of Technology, The Netherlands, 1997.

Hees, M.Th. van, *User guide QUAESTOR, Release 3/1997*, MARIN, The Netherlands, April 1997.

Jane's Information Group Limited, *Jane's all the world aircraft*, UK, yearly published.

Jarmulak, J., *Case-based reasoning for NDT data interpretation*, PhD thesis, Delft University of Technology, The Netherlands, April 1999.

Kolodner, J., *Case-based reasoning*, Morgan Kaufmann Publishers Inc., 1993.

Kolodner, J., *Making the implicit explicit: clarifying the principles of cased-based reasoning*, in: Leake, D.B. (ed.), *Case-based reasoning: experiences, lessons & future directions*, AAAI Press, 1996.

Kroo, I., Takai, M., *A quasi-procedural, knowledge-based system for aircraft design*, AIAA 88-6502/-4428, presented at: *AIAA/AHS/ASEE Aircraft Design, Systems and Operations Meeting*, Atlanta, Georgia, USA, September 1988.

Kroo, I., *An interactive system for aircraft design and optimization*, AIAA 92-1190, presented at: *AIAA 1992 Aerospace Design Conference*, Irvine, CA, USA, Februari 1992.

Kroo, I., Altus, S., Braun, R., Gage, P., Sobieski, I., *Multidisciplinary optimization methods for aircraft preliminary design*, AIAA 94-4325/-2543, in: *5th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, Panama City Beach, FL, USA, September 1994.

Leake, D.B., *Case-Based Reasoning: issues, methods, and technology*, tutorial for: *The First International Conference on Case-based Reasoning*, Sesimbra, Portugal, October 1995.

Leake, D.B., *Case-Based Reasoning: experiences, lessons, and future directions*, AAAI Press, 1996.

Logan, B., Smithers, T., *Creativity and design as exploration*, in: Gero, J.S., Maher, M.L. (eds), *Modelling creativity and knowledge-based creative design*, Lawrence Erlbaum Associates Inc., 1993.

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

Maher, M.L., Pu, P. (eds), *Issues and applications of base-based reasoning in design*, Lawrence Erlbaum Associates Inc, 1997.

Mason, W.H., Arledge, T.K., *ACSYNT aerodynamic estimation- an examination and validation for use in conceptual design*, AIAA 93-0973, presented at: *AIAA/AHS/ASEE Aerospace Design Conference*, Irvine, CA, Februari 16-19, 1993.

Miles, J., Moore, C., *Practical knowledge-based systems in conceptual design*, Springer-Verlag, 1994.

Mittal, S., Araya, A., *A knowledge-based framework for design*, in: Tong, Chr., Sriram, D. (eds), *Artificial Intelligence in Engineering Design*, Volume 1, 1992, pp. 273-293.

Netten, B.D., *Knowledge based conceptual design; an application to fibre reinforced composite sandwich panels*, PhD thesis, Delft University of Technology, The Netherlands, Februari 1997.

Netten, B.D., Vingerhoeds, R.A., *EADOCS: conceptual design in three phases - An application to fibre reinforced composite panels*, in: *Engineering Applications in Artificial Intelligence*, Vol. 10, No. 2, 1997, pp. 129-138.

Olsen, T.D., *Aircraft design tools for PC's*, SAE 971474, presented at: *General, Corporate and Regional Aviation Meeting and Exposition*, Wichita, Kansas, USA, April 1997.

PACE Aerospace Engineering and Information Technology GmbH, Germany, http://www.pace.de, 2000.

Pasaribu, H.M., *An approach to configuration design synthesis of subsonic transport aircraft using Artificial Intelligence techniques,* PhD thesis, Cranfield Institute of Technology, UK, December 1991.

PTC, *Pro/ENGINEER manuals, release 2000i*, Parametric Technology Corporation, USA, 1999.

Pugh, S., *Creating innovative products using total design; the living legacy of Stuart Pugh*, Addison-Wesley Publishing Company, 1996.

Raymer, D.P., Albrecht, S.K., *CDS - the designer's media, the analyst's model,* ICAS-82-1.7.2, 1982, pg. 1155-1163.

Raymer, D.P., *Aircraft design: a conceptual approach*, AIAA education series, 1992.

Raymer, D.P., *RDS: a PC-based aircraft design, sizing and performance system,* AIAA 92-4226, presented at: *AIAA Aircraft Design Systems Meeting*, Hilton Head, South Carolina, USA, August 1992.

Raymer, D.P., *Aircraft design optimization on a personal computer*, AIAA 96-5609, presented at: *1996 World Aviation Congress*, Los Angelos, CA, USA, October 1996.

Rentema, D.W.E., Jansen, F.W., Netten, B.D., Vingerhoeds, R.A., *An AI-based support tool for the conceptual design of aircraft*, presented at: *Computer Aided Conceptual Design (CACD'97); 1997 Lancaster International Workshop on Engineering Design*, Cumbria, England, April 1997.

Rentema, D.W.E., Jansen, F.W., Torenbeek, E., *The application of AI and geometric modelling techniques in conceptual aircarft design*, AIAA-98-4824, presented at: *7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, St. Louis, Missouri, USA, September 1998.

Roozenburg, N.F.M., *On the pattern of reasoning in innovative design*, in: *Design Studies*, Vol. 14, No.1, Januari 1993, pp. 4-18.

Rosenman, M.A., Gero, J.S., *Creativity in design using a design prototype approach*, in: Gero, J.S., Maher, M.L. (eds), *Modelling Creativity and Knowledge-based Creative Design*, Lawrence Erlbaum Associates Inc., 1993.

Rosenman, M.A., Gero, J.S., *The what, the how and the why in design*, in: *Applied Artificial Intelligence*, Vol. 8, 1994, pp. 199-218.

Roskam, J., *Airplane design; parts I to VIII*, Roskam Aviation and Engineering Corporation, Ottawa, Kansas, USA, 1987-1990.

Roskam, J., Malaek, S., *Automated aircraft configuration design and analysis*, SAE 891072, presented at: *General Aviation Aircraft Meeting and Exposition*, Wichita, Kansas, USA, April 1989.

Ruijgrok, G.J.J., *Elements of airplane performance*, Delft University Press, The Netherlands, 1994.

Russell, S., Norvig, P., *Artificial Intelligence; a Modern Approach*, Prentice Hall, 1995.

Schank, R.C., Abelson, R.P., *Scripts, plans, goals and understanding*, Erlbaum, 1977.

Schipperijn, B., *User interface and geometrical module of the AIDA system*, MSc thesis, Delft University of Technology, The Netherlands, 1998.

Torenbeek, E., *Synthesis of subsonic airplane design*, Kluwer Academic Publishers, Dordrecht, The Netherlands, 1982.

Xie, X., Haberland, Ch., *A new numerical design tool for concept evaluation of propeller aircraft*, in: *Aircraft Design*, Vol. 2 nr. 4, Elsevier Science Ltd, 1999, pg. 147-165.

Yuan, L., *A parametric approach to preliminary design for aircraft and spacecraft configuration*, ICAS-92-7.2.1, in: *18th Congress of the International Council of the Aeronautical Sciences*, Beijing, People's Republic of China, September 1992.

Acknowledgements

Having finished this thesis, I would like express my appreciation for the people who have supported me. First of all I thank the persons who have initiated this project: Rob Vingerhoeds and Bart Netten. With their experience in the field of applied Artificial Intelligence techniques they have also been very supportive in the first period of the project, when they were still employed at Delft University of Technology. In this period I also got much assistance from Jan Middel with regard to aircraft design issues. I appreciate the discussions we had in our periodic meetings together with professor Henk Koppelaar and professor Waltraud Gerhardt. I am also much obliged to Martin van Hees, who was so kind by letting me use his Rule-Based Reasoning tool Quaestor. But most gratitude I am obliged to my promoters, who supported me over all these years. Especially the stimulating discussions and encouragements of professor Erik Jansen has helped me through the process of writing this thesis.

I also want to thank the people of the three involved university sections for their hospitality, through the years of reorganisations and migrations. I had a great time thanks to the good atmosphere. In that respect I want to mention in particular the people with whom I shared the rooms with for many years: Bart Netten and Leon Rothkrantz of the Knowledge Based Systems section and Michiel Haanschoten of the Aircraft Design section.

And last but not least I want to give thanks to all my friends and family, for being there and for their motivation to finish this thesis, in one way or another.

October 2004, Eindhoven.

Date Rentema

Curriculum Vitae

Date Rentema was born in Soest, The Netherlands, on the 25th of September 1966. In 1984 he completed the secondary school "Het Baarnsch Lyceum". After graduating at the Aeronautical Department of the "HTS Haarlem", he received his BSc certificate in 1988. In 1994 he graduated at the Delft University of Technology, section Aerodynamics of the Faculty of Aerospace Engineering, on the subject of experimental and theoretical research on the aerodynamic interaction between propeller and wing.

In November 1994 he started his doctoral research program at the same university. The program, a so-called "Beek" project sponsored by the university to stimulate innovative multi-disciplinary research, involved close cooperation between three sections: the Aircraft Design section of the Faculty of Aerospace Engineering, the Knowledge Based Systems section of the Faculty of Computer Science and the Computer Graphics & CAD/CAM section of the same faculty. The research program resulted in the presented thesis, in 2004.

Since April 2000 the author is employed at DAF Trucks in Eindhoven, The Netherlands, as a research engineer in aerodynamics.

Stellingen behorend bij het proefschrift
"AIDA: Artificial Intelligence supported conceptual Design of Aircraft"
Date Rentema, 15 november, 2004.

1. De kracht van Artificiële Intelligentie (AI) technieken bij het voorontwerp van bijvoorbeeld vliegtuigen ligt in het hergebruik van ontwerpkennis en ervaring (dit proefschrift).

2. De ontwerpcyclus kan worden ondersteund door een combinatie van Case-Based Reasoning, Rule-Based Reasoning en Geometric Reasoning (dit proefschrift).

3. Bij Case-Based Reasoning (CBR) hoeft in de case-data in principe geen onderscheid gemaakt te worden tussen het type data, zoals de categorieën Function, Behaviour of Structure.

4. De uitspraak "de uitzondering bevestigt de regel" geeft goed aan hoe moeilijk het is om een algemeen geldende regel op te stellen.

5. Uit het dagelijkse gebruik om zaken uit te leggen met behulp van gelijksoortige voorbeelden, blijkt hoe krachtig het redeneren naar analogie is.

6. Statistische data wordt vaak misbruikt om aan een eigen gedachtengang de schijn van objectiviteit te geven.

7. Niet het hebben van vooroordelen is erg, maar het beschouwen van vooroordelen als de absolute waarheid.

8. Het gebruik van regels, stellingen of citaten dient gewantrouwd te worden, als er niets over de oorspronkelijke context of aanleiding vermeld wordt.

9. Het opdelen van data in de types Function, Behaviour en Structure is ook voor het ontwerp proces van trucks uitermate verhelderend.

10. Hoe intelligenter de apparaten, hoe meer intelligentie de gebruiker nodig lijkt te hebben.

Deze stellingen worden verdedigbaar geacht en zijn als zodanig goedgekeurd door de promotoren.

Propositions accompanying the thesis
"AIDA: Artificial Intelligence supported conceptual Design of Aircraft"
Date Rentema, November 15th, 2004.

1. The power of Artificial Intelligence (AI) techniques for the conceptual design of for example aircraft, is in the reuse of design knowledge and experience (this thesis).

2. The design cycle can be supported by the use of a combination of Case-Based Reasoning, Rule-Based Reasoning and Geometric Reasoning (this thesis).

3. Within Case-Based Reasoning (CBR) there is, in principle, no need for discrimination in the case data between data types, such as the categories Function, Behaviour and Structure.

4. The general saying "every rule has an exception" illustrates the difficulty in specifying general rules.

5. The prevalence of using examples in explanations indicates the power of reasoning by analogy.

6. Statistics are often abused to give an opinion the appearance of objectivity.

7. Prejudice is not bad, but holding it for the absolute truth is.

8. Care should be taken with the use of rules, propositions and citations when the original context is not mentioned.

9. The classification of data into the categories Function, Behaviour and Structure is also clarifying in the design process of trucks.

10. The more intelligent the appliance, the more intelligence the user seems to need.

These propositions are regarded as defendable, and have been approved as such by the promotors.