11. An AI Tool for Conceptual Design of Complex Products

Date Rentema and Erik Jansen

Faculty of Information Technology and Systems
Faculty of Aerospace Engineering
Delft University of Technology

Abstract

This paper describes the set-up of AIDA, a computer tool that supports the conceptual design of complex objects and products. The AIDA system is based on a combination of three methods: Case Based Reasoning for suggesting initial proposals, Rule-Based Reasoning to assess these proposals on their functional merit, and Constraint-Based Geometric Modelling to model and visualise the proposals. These reasoning and modelling techniques are implemented in separate modules. In this paper we present the underlying ideas of the AIDA system and show how this framework can be used for conceptual design of aircraft.

11.1 Introduction

Aircraft design is a complex problem, comparable with the design of other complex objects and systems like buildings, industrial products, and ships. The list of requirements for an aircraft design may contain a large number of issues, from functional requirements such as the aircraft's size, capacity, and payload-range ratio, 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 the 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 1986), Rds (Raymer 1992) and Adas (Bil 1988, 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 detailed product, only to learn that the proposed solution in the end does not satisfy the functional requirements.

Another drawback of the 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 the physics and these experiences, which results in lists of advantages and disadvantages coupled to each configuration choice. These lists are dominated by qualitative arguments that make 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 the case of aircraft: the fuselage, the wing, the tail-surfaces, the undercarriage, and the powerplant. Because conceptual design only deals with approximate sizes, this 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 paper, we propose a new design methodology for conceptual design, using several AI techniques, which, in combination, support the total design cycle of proposing and evaluating solutions. First we discuss conceptual design and how AI techniques can be used to implement the required qualitative kind of reasoning needed for suggesting and analysing design concepts. Then we describe the implementation of AIDA and show how it can be used to design aircraft.

11.2 Conceptual design

11.2.1 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 the designer 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 that is visualized in Figure 11.1.

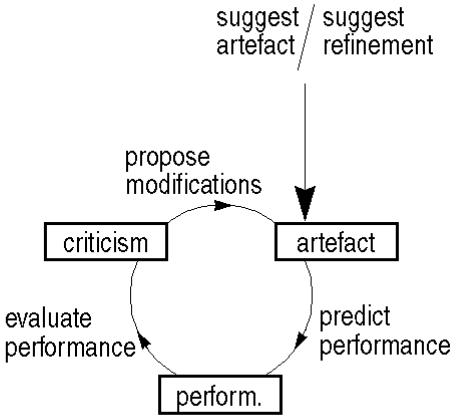

Figure 11.1: The basic tasks in the design cycle.

Using his experience, the designer first suggests a solution, then the performances are predicted (because a full assessment is seldom possible), then the performances are evaluated with respect to the requirements, then modifications are proposed to remedy the shortcomings, and the cycle repeats. When the solution is deemed to be good enough, the

designer suggests refinements. This means that the artefact will be described more completely by adding details, for example by focusing on one of the components.
An often-applied strategy to avoid exhaustive generate-and-test searches 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 to 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.

11.2.2 AI and design

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 cannot 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 cannot be applied to the design problem because the solution space is sparse and discontinuous. The third form of reasoning (reasoning by analogy) is best exemplified by Cased-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 optimisation. 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 11.2.

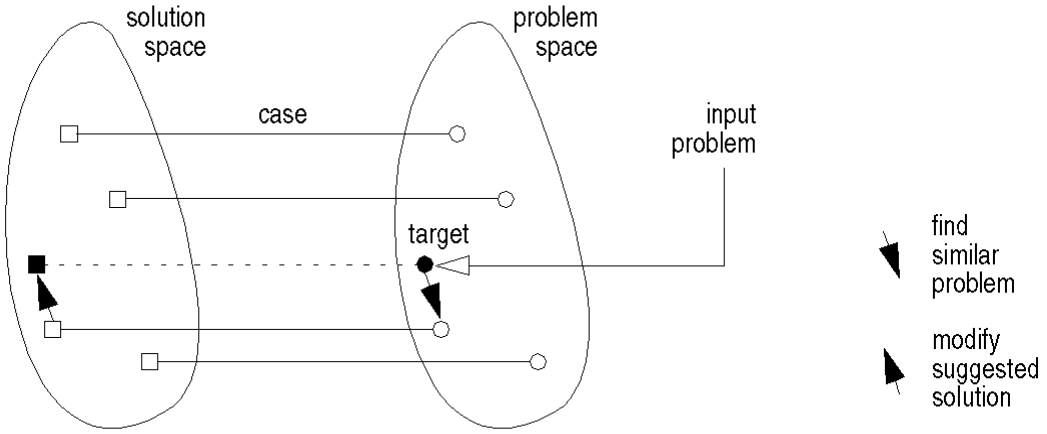

Figure 11.2: Principles of problem-solving with Case-Based Reasoning (from Leake 1996).

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 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 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 for the behaviour and structure of the design. Some functionality may therefore be an indirect result of behavioural and structural choices given by the list of requirements, but not directly described in the list of requirements.

For a proper evaluation of the structure via the behaviour and the function components, explicit knowledge can be applied. Because of 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, rule-based techniques 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 parameterisation of the structure, and/or are 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.
In the next section we will describe how these techniques can be applied for conceptual design. A system set-up is presented based on the use of case-based reasoning to propose design concepts, rule-based-techniques for the functional evaluation, and constrained-based geometric modelling for the geometrical evaluation of these concepts.

11.3 The AIDA system

The AIDA system (AI-supported Design of Aircraft) consists of three modules and a central interface; see Figure 11.3:

- Case-based reasoning (CBR) module. In this module case-based reasoning techniques are implemented to generate an acceptable initial concept that can be modified in the Functional module.
- Functional module. In this module rule-based reasoning techniques are implemented to perform sensitivity studies on the primary parameters of the concept. With these studies a feasible concept is designed.
- Geometrical module. This module automatically models and visualizes the concept. It uses feature-based and constraint-based modelling techniques.
- Central user interface (CUI). This module handles the communication between the three modules.

The following paragraphs describe the separate modules. For each module, existing tools have been used.

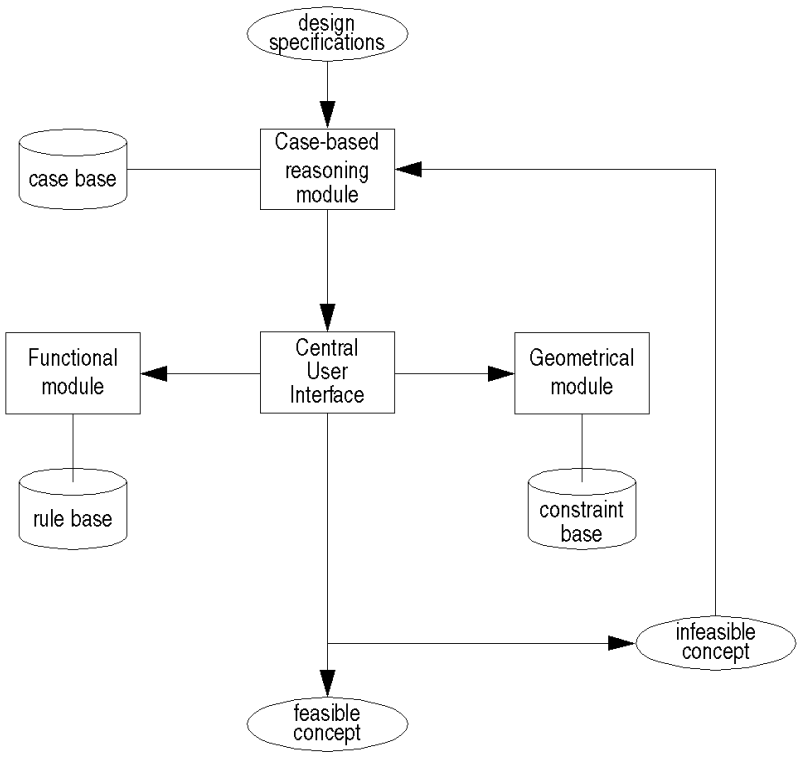

Figure 11.3: Modular set-up of the AIDA-tool.

11.3.1 Case-Based Reasoning module

As suggested by its name, the Case-based reasoning (CBR) module applies case-based reasoning techniques to generate an acceptable concept from the design specifications. These techniques enable the use of the design experience that is implicitly available in existing cases. Also, case-based reasoning is an approach to learning, since the result of previous design sessions can be added to the case-base, making it available for future design problems.
A complete case-based reasoning process can be considered as a cycle of four sequential steps; see (Aamodt 1994) and Figure 11.4:

- Retrieve: Find cases in the case-base which resembles the problem description;
- Reuse: Copy case-data or combine data of more cases;
- Revise: Evaluate the proposed solution; and
- Retain: Put successful 'learned case' in the case-base.

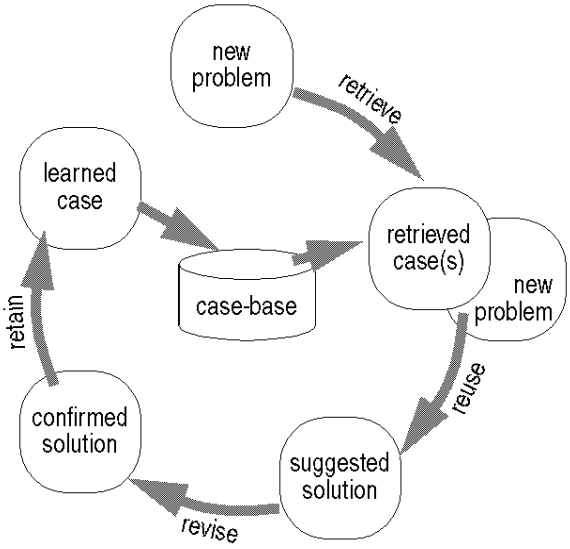

Figure 11.4: The CBR cycle (adapted from Aamodt 1994).

The problem description defines the 'new case'. In the Retrieve step the case-base is searched for cases with data matching the 'new case'. The cases with most similar data are retrieved. In this step the matching process is most critical.
In the Reuse step, data is copied from a 'retrieved case'. Usually, the 'retrieved case' does not completely match the 'new case', i.e. the best matching case does not completely solve the problem. In that situation the data of more 'retrieved cases' can be combined. In other words, the best matching case is changed/adapted with data of other selected cases. This adaptation process requires domain knowledge and is very complex.
The result of the Reuse step, the 'solved case', is a suggested solution to the problem. It is evaluated and repaired when necessary in the Revise step. The evaluation process is often performed by numerical tools. This process also requires domain knowledge, as does the repair process. The result is a 'tested/ repaired case', or a confirmed solution to the problem.
The learning aspect is implemented by adding information about the confirmed solution to the case-base. The Retain step handles the transformation from the 'tested/repaired case' into the 'learned case'.
In AIDA, only the Retrieve, the Reuse and the Retain steps are implemented in the Case-based reasoning module. The evaluation in the Revise step is taken care of by the Functional

module, using rule-based reasoning techniques; see next paragraph. The CBR module has been developed in Eadocs (Netten 1997, 1998), a design system for composite sandwich panels. In this CBR module the Retrieve step, as well as the Reuse and the Retain steps have been implemented; the Revise step has been implemented in another module.
In aircraft design, the cases contain data about their function or performances, such as the range and speed, and data about their structure or physics, such as the weights and sizes. Also data which numerically expresses the quality of the aircraft is added, such as the wing-loading and maximum lift-coefficient; according to (Rosenman 1994) these can be considered as behavioural data.
The CBR module consists of two parts. The first part generates an indexing network. This network provides an index to the cases by the use of parameter domains. These parameter domains allow qualitative labelling of the continuous parameter values, such as 'small' and 'moderate' etc., to enable a kind of qualitative matching. The network is created off-line to improve the efficiency.
The second part uses the network to search for cases similar to the specified 'target set'. This is done on-line. For each part of the target set the matching results are shown, and the cases are ranked accordingly. The importance of each part of the target set is given by priority-values. Figure 11.5 shows a list of best-matching aircraft configurations.

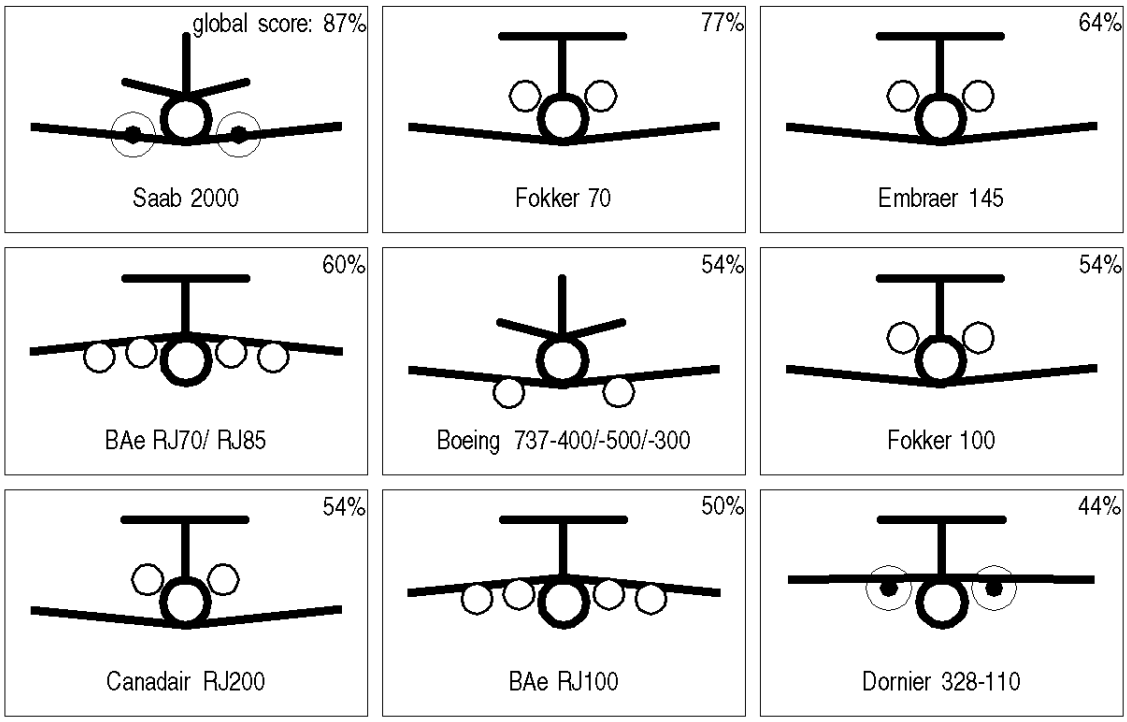

Figure 11.5: An example of best-matching aircraft configurations.

The strategy used for the CBR module is to first retrieve a 'best-matching' case completely, and then to reuse parts of other cases for the adaptations. The adaptation process, however, disrupts the link between the functional and the structural data. Therefore the adapted case is probably not valid anymore. The adapted case should be evaluated before other adaptations can be usefully applied.

It is difficult to adapt the case properly, due to the many interactions between the functional data and the structural data. Therefore a strategy is followed which should lead to as little adaptations as possible. A secondary target set is defined, consisting of the rest of the specifications that the ‘best-matching’ case does not satisfy, together with the most important structural and behavioural data of the ‘best-matching’ case. The result of the new matching process will give cases that resemble the structure of the ‘best-matching’ case, and which resemble the functional data as indicated by the unsatisfied specifications.
Domain knowledge is used to support the adaptation process. Expert rules have been collected which may help with focusing on the relevant data. For instance, when the ‘best-matching’ case does not reach the specified speed (function), the designer should focus on the thickness of the wing and its sweepback angle (structure). In Figure 11.6 this strategy is summarized.

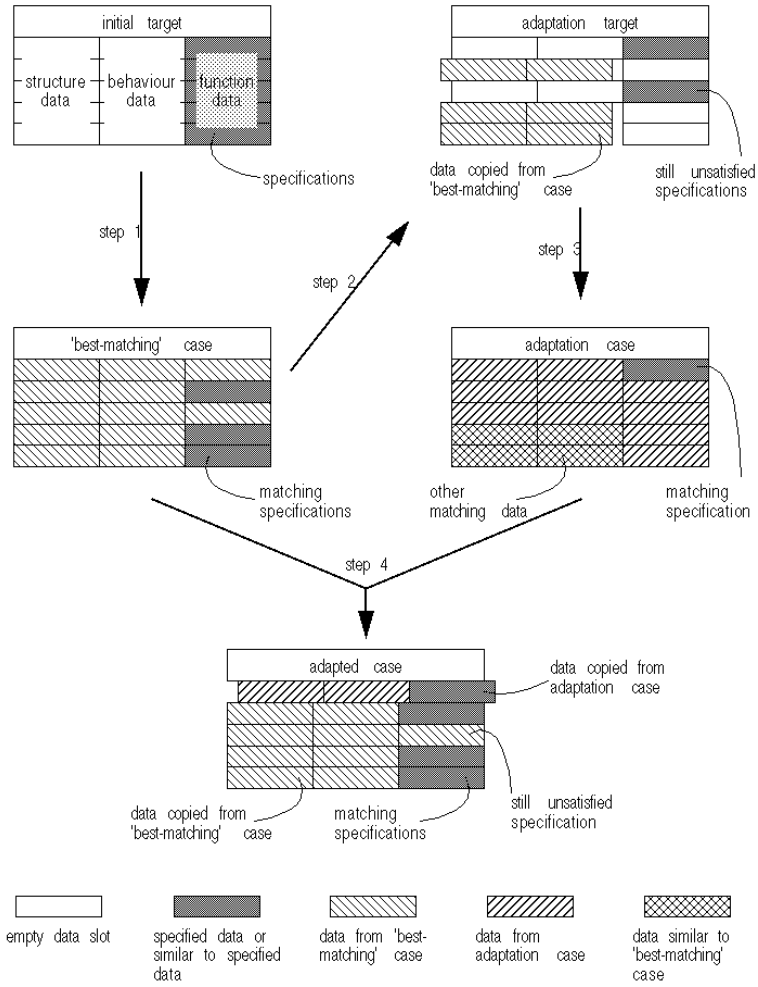

Figure 11.6: Case adaptation strategy.

The CBR module delivers a complete case, which may be inconsistent due to the adaptations. The complete structural data set is required by the Geometrical module in order to make a solid model of the concept. Some structural and some behavioural data is required by the Functional module when generating a relational network. They provide proper initial estimations that can be evaluated and modified.

11.3.2 Functional module

The Functional module supports the execution of parameter studies. These are used to evaluate and modify the adapted case as produced by the CBR module. The result is a feasible concept, according to the available knowledge. This knowledge is represented by numerical relations which express the heuristics such as collected in the books of Roskam (Roskam 1987) and Torenbeek (Torenbeek 1982).

In the Functional module rule-based reasoning techniques are applied to generate a network of numerical relations. The rules implement heuristic knowledge and simplified physics in an algebraic format. The network of these rules links functional parameters (from the specifications) to structural parameters (from the designed object). With this network sensitivity studies can be performed to estimate the structural parameter-values. This reasoning technique offers the flexibility to easily generate a different network for a different set of specifications.

The Functional module is implemented in Quaestor (van Hees 1997), an expert governed system for the assembly, execution and maintenance of revise parametric design models. This tool has been developed to support ship design in the early phases of design by improving access to and control over design-related knowledge.

Before Quaestor can be used, the rule-base has to be built. The rules represent numerical relations in an algebraic format. To each rule, conditions can be added to incorporate its limited validity. This is often the case with the heuristics and simplified physics. Within Quaestor the conditions are called 'constraints'. The third type of element within Quaestor is the parameter. The rule-base is built by one or more experts, and does not have to include the designer that uses the Functional module.

The designer starts a Quaestor session selecting the parameters to be calculated, for example the specification parameters. These are called the goal parameters. The inference engine of Quaestor searches the rule-base for relations that are associated with the goal parameter. That is, the parameter is one of the variables of the numerical relation. So, the backward chaining strategy is applied.

The designer selects one of the suggested relations. Usually the new relation introduces other, unknown parameters. The designer can make the unknown independent or dependent. The parameter becomes independent when it gets a value, for instance the value generated by the CBR module. The parameter becomes dependent when it is assigned as a new goal parameter for which other relations have to be found. The creation of the relational network is finished when no goal parameters are left. When the network is completed, the number of dependent parameters is equal to the number of relations. Quaestor uses Newton-Raphson and simplex methods to solve the set of relations, and calculate the goal parameters. Figure 11.7 shows a simple numerical link.

It is important to notice that Quaestor also searches for relations with the goal parameter on the right-hand side of the algebraic notation. This makes Quaestor very flexible. Instead of an algebraic relation, an external application can also be part of the network. This possibility makes the module even more powerful. However, the input and output parameters of these applications have to be known in advance.

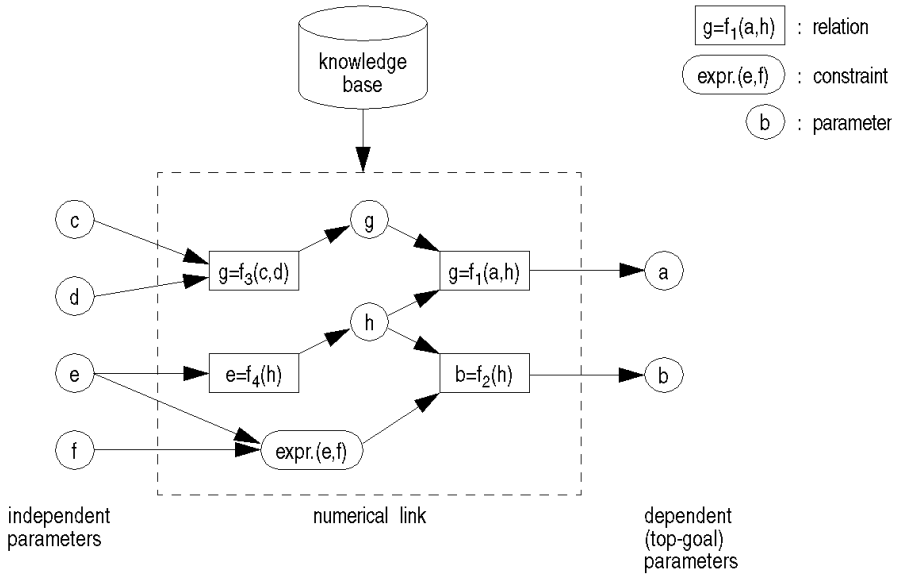

Figure 11.7: A Quaestor template, linking independent with dependent parameters.

To perform parameter studies, independent parameters can be given an array of values. The resulting arrays of the goal parameters can be saved in an ASCII-file, to be viewed graphically by the Central user interface.

11.3.3 Geometrical module

The Geometrical module models and visualizes the suggested aircraft concept automatically. This gives the designer a visual feedback of the concept. The module can also deduce some typical geometrical information, such as volumes and areas. The solid model can be used as input for other computer programs.

The module is implemented in Pro/Engineer (PTC 1999), a commercial feature based solid modeller. In this modeller the solids are constructed with engineering features, such as holes, protrusions, rounds etc. Features can be 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.

Pro/Engineer can handle the variation of continuous parameters very well. However, to change the definition of features, parts and assemblies requires knowledge of the package, which is not the intention of the Geometrical modeller. Therefore the aircraft concept model is constructed with pre-defined parts, such as the fuselage, the wing, the engines, the horizontal and the vertical tail-surfaces. The geometrical constraints between these parts keep them properly located.

It is possible to visualize different types of these pre-defined parts, for example turbofan and turboprop engines. Both types are pre-defined, and the parameter which defines the type of engine will automatically suppress the other types. In this way the variation of the discrete parameters is managed.

This strategy of using standard aircraft parts is practicable because it is known which configuration elements are in the case-base. Figure 11.8 shows the hierarchical structure of the aircraft model.

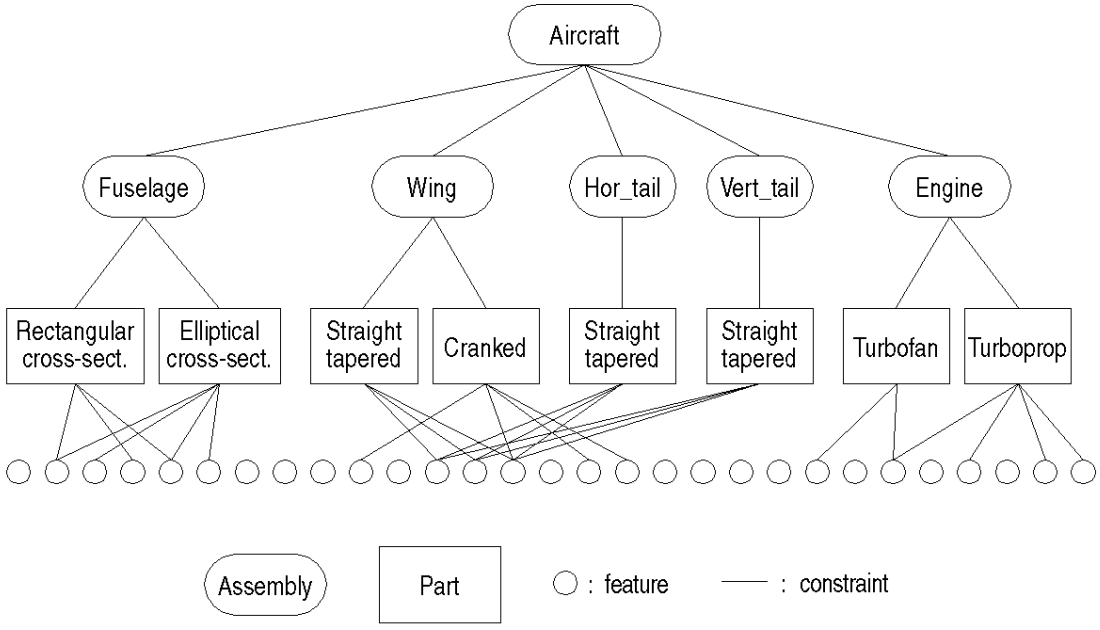

Figure 11.8: The hierarchical structure of the aircraft.

The Geometrical module is able to show a three-dimensional view as well as three side-views of the model. Figure 11.9 gives an example of a three-dimensional view. The model can also be saved in several graphical formats, for use in other computer programs.

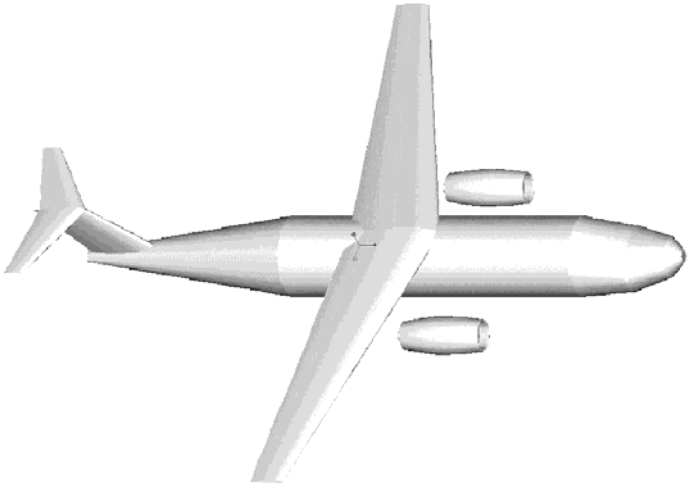

Figure 11.9: A shaded, 3-dimensional view of a concept design.

11.3.4 Central user interface

The Central user interface (CUI) handles the interaction between the other three modules and the designer. It reads the data generated by the CBR module, of the adapted case, filters it and directs the required data to the Geometrical module. The CUI is also able to present the results of the sensitivity studies, performed by the Functional module, in a graphical format. See Figure 11.10 (next page).

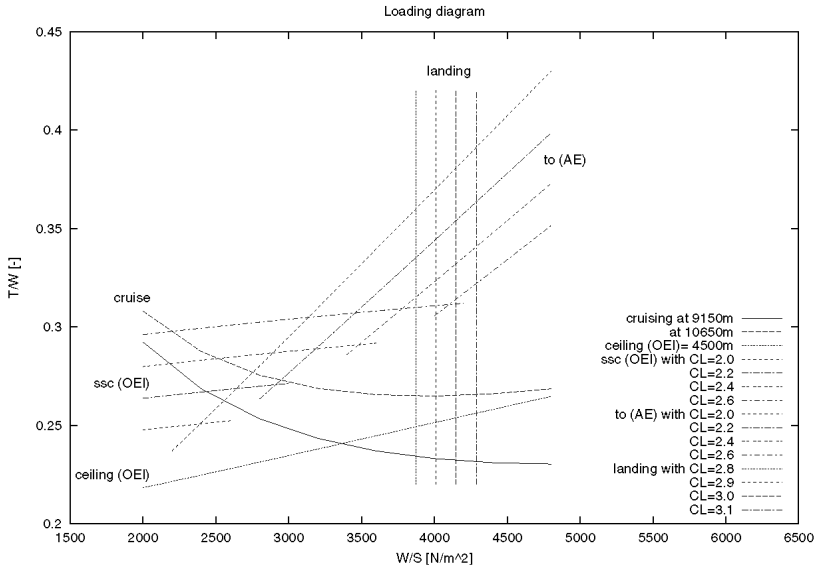

Figure 11.10: Sensitivity studies of sime. Behaviour parameters in aircraft conceptual design.

11.4 Conclusion

We have analysed the conceptual design process, and suggested a design cycle that uses CBR-techniques to propose and adapt initial concepts, RBR-techniques to analyse and evaluate the concept, and geometric modelling techniques that model the concept automatically. These three techniques are implemented in three independent modules, with a central user interface to connect the modules.

The system has been evaluated for the conceptual design of aircraft. This application allows the decomposition of the design product into basic components. The current approach therefore relies heavily on the decomposition of the design product into basic components. For a more general approach without the basic components the case representation will need to be more comprehensive.

Another aspect of further research will be the implementation of the separate modules into one integrated framework.

11.5 References

Aamodt, A. and Plaza, E. (1994). Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches, in: *AICOM*, Vol. 7, Nr. **1**, March.

Bil, C. (1989). ADAS: A Design System for Aircraft Configuration Development, *AIAA 89-2131*.

Bil, C. (1988). *Development and Application of a Computer Based System for Conceptual Aircraft Design*, PhD thesis, Delft University of Technology, The Netherlands.

Gero, J.S. and Tham, K.W. and Lee, H.S. (1992). Behaviour: A Link Between Function and Structure in Design, in: Brown, D.C. and Waldron, M. and Yoshikawa, H. (eds), *Intelligent Computer Aided Design*, Elsevier Science Publishers B.V., The Netherlands, pp. 193-221.

Hees, M van (1997). *Quaestor: Expert Governed Parametric Model Assembling*, PhD thesis, Delft University of Technology, The Netherlands.

Leake, D.B. (1996). *Case-Based Reasoning: Experiences, Lessons, and Future Directions*, AAAI Press.

Mason, W.H. and Arledge, T.K. (1993). Acsynt Aerodynamic Estimation - An Examination and Validation for Use in Conceptual Design, *AIAA 93-0973*.

Mittal, S. and Frayman (1989). Towards a Generic Model of Configuration Tasks, *Proc. 11th IJCAI*, Morgan Kaufman, pp. 1395 - 1401.

Netten B.D. and Vingerhoeds R.A. and Koppelaar, H. (1995). Expert Assisted Design of Aircraft, in: Oxman R.M. and Bax M.F.Th and Achten H.H. (eds), *Design Research in The Netherlands*, Eindhoven University of Technology, pp. 63-67.

Netten, B.D. (1997). *Knowledge Based Conceptual Design; An Application to Fibre Reinforced Composite Sandwich Panels*, PhD thesis, Delft University of Technology.

Netten, B.D. (1998). A Hybrid Reasoning System for Conceptual Design, in: *AAAI-98 Workshop Case-Based Reasoning Integrations*, Madison, Wisconsin.

Oxman, R.M. and Bax, M.F.Th. and Achten, H.H. (1995). *Design Research in the Netherlands – A Symposium Convened by Design Methods Group and Information Technology for Architecture*, January 1995. Bouwstenen 37. Faculteit Bouwkunde, Eindhoven University of Technology.

PTC. (1999). *Pro/Engineer Manuals, Release 2000i*, Parametric Technology Corporation, USA.

Raymer, D.P. (1992). RDS: A PC-Based Aircraft Design, Sizing and Performance System, *AIAA 92-4226*.

Rentema D.W.E. and Vingerhoeds R.A. and Netten B.D. and Jansen F.W. (1996). The Development of a Support Tool for the First Phases of Aircraft Design, in: *ICTAI96 Workshop on Artificial Intelligence for Aeronautics and Space*, Toulouse, France.

Rentema D.W.E. and Jansen F.W. and Netten B.D. and Vingerhoeds R.A. (1997). An AI-Based Support Tool for the Conceptual Design of Aircraft, in: *Computer Aided Conceptual Design (CACD'97)*, Cumbria.

Rentema D.W.E. and Jansen F.W. and Torenbeek, E. (1998). The Application of AI and Geometric Modelling Techniques in Conceptual Aircraft Design, in: *7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, St.Louis, Missouri.

Rosenman, M.A and Gero, J.S. (1994). The What, the How, and the Why in Design, in: *Applied Artificial Intelligence*, Vol. **8**.

Roskam, J. (1987-1999). *Airplane Design; Parts I-VIII*, Roskam Aviation and Engineering Corporation, Ottawa, Kansas.

Roskam, J. and Anemaat, W.A. (1996). General Aviation Aircraft Design Methodology in a PC Environment, *AIAA 96-5520*.

Torenbeek, E. (1982). *Synthesis of Subsonic Airplane Design*, Kluwer Academic Publishers, Dordrecht.