The process of design aircraft systems is becoming more and more complex, due to an increasing amount of requirements. Moreover, the knowledge on how to solve these complex design problems becomes less readily available, because of a decrease in availability of intellectual resources and reduced knowledge transfer opportunities. Aerospace companies need to capitalise on the knowledge available within their companies, in order to deal with the challenges of increasing complexity and competition.

The research presented in this thesis contributes to tackling the above challenges. A knowledge based method for solving complex detailed design problems is presented. The process of setting-up a solution finding approach is discussed by means of a design problem in the detailed design of fibre metal laminate (FML) fuselage panels. The principles of knowledge based engineering (KBE) are used to setup a software application for finding solutions to the detailed design problem in FML.

A method for solving complexity in design problems is presented. The method uses a structured approach of transforming the real world problem, via an expert view on the problem, to a mathematical model of the problem. Finally a solution finding strategy is tailored for the problem at hand, by combining available solution finding knowledge with expert problem solving knowledge from different knowledge domains.

ISBN 978-9-0902182-3-6

TU Delft
Delft University of Technology

STORK Fokker
Stork Fokker Aeropace

B. Vermeulen Knowledge Based Method for Solving Complexity in Design Problems

KNOWLEDGE BASED METHOD FOR SOLVING COMPLEXITY IN DESIGN PROBLEMS

B. Vermeulen

B. Vermeulen Knowledge Based Method for Solving Complexity in Design Problems

Promotion of B. Vermeulen

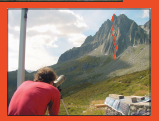
Wednesday June 20th, 2007

Introduction 12:00-12:25

Defence 12:30-13:30

Reception 14:00-15:30

Dinner & Drinks 17:30-24:00



KNOWLEDGE BASED METHOD FOR SOLVING COMPLEXITY IN DESIGN PROBLEMS

KNOWLEDGE BASED METHOD FOR SOLVING COMPLEXITY IN DESIGN PROBLEMS

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 voor Promoties,
in het openbaar te verdedigen op 20 juni 2007 om 12:30 uur

door

Brent VERMEULEN
ingenieur in de luchtvaart en ruimtevaart
geboren te Woerden

Dit proefschrift is goedgekeurd door de promotoren:
Prof. dr. ir. M.J.L. van Tooren
Prof. dr. ir. R. Benedictus

Samenstelling promotiecommissie:

Rector Magnificus	voorzitter
Prof. dr. ir. M.J.L. van Tooren	Technische Universiteit Delft, promotor
Prof. dr. ir. R. Benedictus	Technische Universiteit Delft, promotor
Prof. dr. ir. C. Roos	Technische Universiteit Delft
Prof. dr. Z. Gurdal	Technische Universiteit Delft
Dr. S. Ahmed	Technical University of Denmark
Dr. ir. T.J. de Vries	Airbus Germany GmbH
Ir L.J.B. Peeters	Stork Fokker AESP

ISBN: 978-90-90218-23-6

Keywords: Fibre Metal Laminates, Knowledge Based Engineering, Constraint Programming, Heuristics, Engineering Ontologies

Printed in The Netherlands.

Cover: The problem of climbing a mountain is comprised of analysing the elements of the mountain's surroundings, and setting up a strategy for the ascent. Photos by Dirk Samson: Salbitschijen (2981), Switzerland.

This research has been funded by Stork Fokker Aerospace and
Delft University of Technology

Delft University of Technology
http://www.tudelft.nl

Stork Fokker Aerospace
http://www.stork.com/Aerospace

voor mijn lieve Essie

Contents

Acknowledgements

The support and trust of five persons created the right conditions for the foundation of my research, and my first words of thanks are to these persons. Tjerk de Vries, Leo Peeters and Robert Jan de Boer, thank you for your vision and help in setting-up this research. Jan de Jonge and Michel van Tooren, thank you for your decisiveness, which resulted in the long awaited start of the research.

My research was supported by two research groups of Delft University of Technology: *Aerospace Materials* and *Design, Integration and Operation of Aircraft and Rotorcraft*. A multi-disciplinary committee was formed, in which I presented my developments every two months. I truly appreciated these sessions, during which we discussed both my research and the latest developments at Airbus, Fokker and TU Delft. Michel helped me in formalising my line of reasoning (that seemed so obvious to me!). Leo represented the interests of Stork Fokker AESP by addressing the practical implication of the research. Tjerk provided a neutral view on the progress, being able to both use the Airbus and TU Delft 'hat'. Jos Sinke repeatedly provided me with practical how-to-do-research advice. Rinze Benedictus, being the new professor of *Aerospace Materials*, helped me to better formulate and defend my work by our every now and then philosophical discussions on my research.

I would like to thank my roommates at TU Delft, Mario Vesco and later on Rik Jan Lemmen, who showed me that my PhD-struggles were nothing but standard procedures; the staff of *Aerospace Materials*, Rene Alderliesten, Johannes Homan, Maarten Bakker and Kees Sudmeijer, for the lunch-time discussions on what the world should look like; the PhD-students at *Design, Integration and Operation of Aircraft and Rotorcraft*, Gianfranco La Rocca, Giampietro Carpentieri, Jochem Berends, Marco Nawijn, Joost Schut and Ton van der Laan, for the helpful domain related discussions; my fellow 'halbewoners' without whom the three o'clock coffee wouldn't have been something to look forward to; the master students I have been privileged to supervise, Martijn, Michel, Joeri en Stefan; and the most important person at the TU Delft, Gemma van der Windt, for the personal talks and interest, yelling across the corridor and always finding the time to help out.

The knowledge of the experts has been invaluable to this research. Therefore I would like to thank the engineers at Stork Fokker AESP, Bart Beusmans, Wilbert Brouwer, Onno Verschoof, Ben Versluis and Max Markestein, for their willingness to share their knowledge with me, despite the prospect of losing their job while doing so. You were a great team and I fully enjoyed the on and off the job time spent together. Besides Stork Fokker AESP, Airbus engineers also participated: Arthur Tillich and Fred Pellenkoft shared their knowledge on FML design, and gave invaluable feedback during various validation sessions. The beer tastes great in Hamburg as well!

Of course there is live besides work. What is a better way to unbend one's mind than to go out in nature and enjoy freedom while climbing, mountain biking, running, or a combination thereof. The climbing trips with Ruben, Eelco, Michiel van den E, Stefan, Michiel and Maayke or Court, replenished my energy. So does running in the swamps of Delft with MarkO, Dirk, Marjan, Hanneke, Court and, maybe in the future, Nannette. I am very privileged to be able to spend weekends out with my Yeti family.

A special thanks to the family whose love and support I couldn't have done without. My parents were always available to share their knowledge of live with me. Everything in life is grace, but it also takes passion, focus and hard work to make the best of it. It is always a joy to meet Martijn, Judith and big-smile Casper. Make sure that we will have your promotiefeest soon! I loved philosophising with Tycho and Gaby on the deeper things in life, and see how that life flourishes in Kieran and Hannah. Perhaps you all will make a family man out of me one day.

Summary

KNOWLEDGE BASED METHOD FOR SOLVING COMPLEXITY IN DESIGN PROBLEMS

by Brent Vermeulen

The process of design aircraft systems is becoming increasingly complex, due to an increasing amount of requirements that have to be fulfilled. Moreover, the knowledge on how to solve these complex design problems becomes less readily available, because of a decrease in availability of intellectual resources and reduced knowledge transfer opportunities. In order to deal with the challenges of increasing complexity and competition, aerospace companies need to capitalise on the knowledge available within their companies.

The research presented in this thesis contributes to tackling the above challenges. A knowledge based solution method dealing with complexity in finding solutions to detailed design problems is presented. The process of setting-up a solution approach is discussed by means of a design problem in the detailed design of fibre metal laminate (FML) fuselage panels. The principles of knowledge based engineering (KBE) are used to setup a software application for finding solutions to the detailed design problem in FML.

Challenges in detailed design

The iterative character of the design process results in changes that have to be implemented in the design. Because of these changes it is imperative to have control both on the flow of information between disciplines involved in the process, and on change propagation between elements in the aircraft (sub)system. Transferring information is complicated, because disciplines use different product models for their analysis, requiring a redefinition of the information. Furthermore, during each design cycle the time consuming process of generating the required design outputs has to be repeated.

KBE is the science of identifying, capturing, storing and re-using expert knowledge. Once the knowledge on executing a process step is formalised, it can be re-used in a software

application to automate the process step. In a KBE application, using expert product and process knowledge, a generative product model is defined. Generative means that it can instantiate different discipline specific views on the product, depending on the information needed by the discipline for their analysis process. Automatic generation of the design outputs can also be facilitated, providing the knowledge on how to execute this step is made explicit. The possibility to automate time consuming and often error sensitive process steps, will reduce the cost of iterations.

Detailed design problem in FML

The detailed design of FML fuselage panels is governed by a large amount of requirements from different disciplines. Detailed information on how the back-up structure of the fuselage is joined to the laminate is needed, to such an extend that the location of each rivet needs to be checked for compliance with the requirements. Implementing all requirements in a feasible product definition asks for a large knowledge of the engineering principles, and results in an iterative and time consuming process. Expert knowledge on the product, design process and solution funding strategy has been captured in entity and activity diagrams. An entity diagram describes the product elements, the relations between the elements and the constraints that act on each element. An activity diagram can be used to graphically represent the activities in a process, identify the relations between the process steps and the rules that apply to the execution of the process step.

Mathematical model

A well-defined mathematical model of the FML design problem is defined, based on the expert view of the problem. Well-defined means the model consists of at least one objective function, and a sequence of design variables whose solution domain is limited by constraints. The variables in the model determine the laminate built-up, the constraints consist of restrictions on rivet positions, so-called no-riveting areas in the laminate, and constraints on the laminate built-up. The design problem can be typified as a constraint satisfaction problem (CSP), having hard and weighted constraints. A solution to the problem complies with all hard requirements. The objective function of the mathematical model is the summation of the cost of violating the weighted constraints, and should be minimised.

A solution finding algorithm is tailored for the mathematical model, based on heuristic solution finding techniques. Heuristics reduce the complexity of the solution finding process, by limiting the solution domain, without simplifying the problem. Both expert and mathematical solution finding knowledge of the problem is used to limit the solution

domain. The efficiency of the solution finding algorithm, in terms of reduction of the solution finding complexity, is evaluated using four test problems. A significant reduction in the number of evaluation of the algorithm is shown, proving the complexity reduction capability of the algorithm.

Implementation in the engineering world

The knowledge based solution system is based on the concepts in a design and engineering engine (DEE). The DEE concepts are software applications performing the necessary steps in the design process. The process starts at the initiator, who collects the required input information, and defines what the product variables are. Next step is to generate product models and export the product and process information to the discipline specific analysis tools. These analysis tools determine the state of the product in terms of objective function value and constraint satisfaction. An evaluator assesses if the product state is in compliance with the requirements, and if not, a search engine will scan the solution domain and export a best solution to the initiator.

An Automated Detailed DEsign Tool (ADDET), containing the knowledge based solution system, is implemented in the engineering world. The effectiveness of ADDET is evaluated in terms of the possibility to generate solutions to the design problem and reduce design time. It is concluded that it is possible to generate solution to the design problem using ADDET. A reduction in process lead-time of 60% for the detailed design of FML fuselage skins is shown. When implementing ADDET in the design process, this reduction in lead-time will result in a large amount of design output being generated in a relative shorter time span. This could lead to bottlenecks in the other departments, since they should check the design outputs in a shorter time span. A process wide re-design is needed to prevent these bottlenecks from causing inefficiencies and cancel out the time saved by implementing ADDET.

Knowledge based solution method

To be able to re-use the problem solving knowledge described in this thesis, a formal model of the applied method is presented using an ontology. An ontology is an explicit description of concepts in a specific domain, defining a formal domain vocabulary. An ontology defines relations between the concepts in the domain, and adds attributes to further specify a concept, making it a suitable means of capturing and re-use knowledge.

Different types of ontologies are used to describe the method domain. The first type describes the experts knowledge on the problem, called a *problem ontology*. It defines the concepts in the domain, their relations, the task set and the solution domain applicable.

The second type is the inference knowledge, describing rules of inference based on the expert and mathematical knowledge of finding solutions to the problem. The third type describes the knowledge on the sequence of deploying the rules of inference in order to most efficiently find solutions to the problem, contained in a so-called problem solving method (PSM). This level of knowledge can be represented using a *task ontology*.

A problem-independent PSM has been defined for the engineering domain, which is made problem specific by tailoring its solution finding strategy for the problem at hand. The method uses a structured approach of transforming the real world problem, via an expert view on the problem, to a mathematical model of the problem, for which solution finding knowledge is present or can be defined using expert problem solving knowledge from different knowledge domains. Finally, the usefulness of the solution system for solving the real world problem should be assessed, by implementing the system in the real engineering world.

Nomenclature

Symbols

a	distance between rivet position and joggle position
c	free edge distance
$\mathbf{C}$	set of constraints imposed on $\mathbf{X}$
C	constraint imposed on $\mathbf{X}$
d	distance between two adjacent rivets
$\mathbf{D}$	set of domains for $\mathbf{X}$
D	domain for variable X
f	function
h	constraint type
L	length
N	quantity
$NoRivetArea$	area where no rivet can be positioned
P	stringer datum
$\mathbf{P}$	a constraint satisfaction problem
$\mathbf{Q}$	set of domains for $\mathbf{X}$ after constraint propagation and discretisation
$\mathbf{R}$	solution sequence to the decomposed problem
R	rivet location
$\mathbf{S}$	solution sequence to a constraint satisfaction problem
S	element of $\mathbf{S}$
u, v	panel coordinate
x, y, z	cartesian coordinate
$\mathbf{X}$	sequence of design variables
X	design variable
Z	zone location
$\mathbf{Z}$	copy of $\mathbf{Q}$
σ	stress

Indices

$1, 2, 3, 4$	index
bn	blunt notch
C	constraint
$cluster$	design variables in a sub-problem
h	hard constraint
$h1$	constraint type 1, see section 4.1.2
$h2$	constraint type 2, see section 4.1.2
i, j, k, m	index
irb	inter rivet buckling
max	maximum
min	minimum
$pattern$	pattern of rivets
R	rivet
u	u-direction
v	v-direction
var	variable
w	weighted constraint
Z	zone element

Abbreviations

ADDET	Automated Detailed DEsign Tool
API	application programming interface
bn	blunt notch
CSP	constraint satisfaction problem
DEE	design and engineering engine
ES	expert system
FAESP	Stork Fokker aerospace
FEM	finite element modeling
FML	fibre metal laminate
GA	genetic algorithm
GUI	graphical user interface
ICARE	illustrations constraints activities rules entities
irb	inter-rivet buckling

KBE	knowledge based engineering
KBS	knowledge based system
MDO	multidisciplinary design optimisation
MOKA	methodology and software tools oriented to knowledge based engineering applications
MOO	multiple objective optimisation
NR	zone in the laminate where no riveting is allowed
OCSP	over-constraint satisfaction problem
PSM	problem solving method
SQP	sequential quadratic programming
UML	unified modeling language
RF	reserve factor
RP	pattern of rivets at equidistance
UDF	user-defined feature (CATIA V5 entity)
VB	visual basic (object-oriented programming language)

Chapter 1

Introduction

The aerospace industry has entered a phase where the emphasis of aircraft development is on safety, noise reduction, cheaper and faster development[1]. Instead of innovations on the product to improve flight performance such as higher or farther, a process focus is required to achieve the envisioned goals. This process focus is intensified because of the increasing complexity of designing aerospace systems. Figure 1.1 shows an exponential increase in design requirements on aerospace vehicles, from the very first motorised flight till the 1990s.

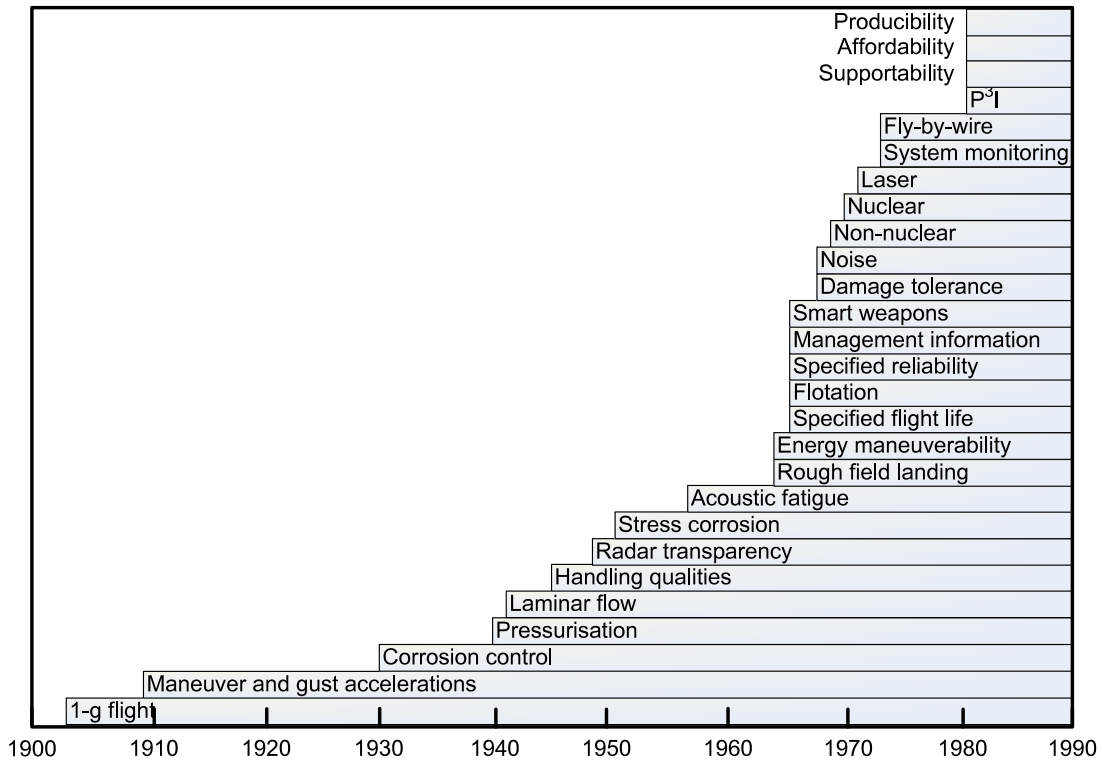


Figure 1.1: Design requirements on aerospace vehicles[2].

Aside from the high goals set for the future, aerospace companies are facing less economic and intellectual resources, increased competition because of globalisation and less knowledge transfer between consecutive programs. This final issue is illustrated in figure 1.2, showing the start of military aerospace projects and a typical career length of an engineer. The same can be identified for the commercial aerospace industry, for instance when looking at the fact that the only new jet powered large transport aircraft developed in the US in the 1990s has been the Boeing 777.

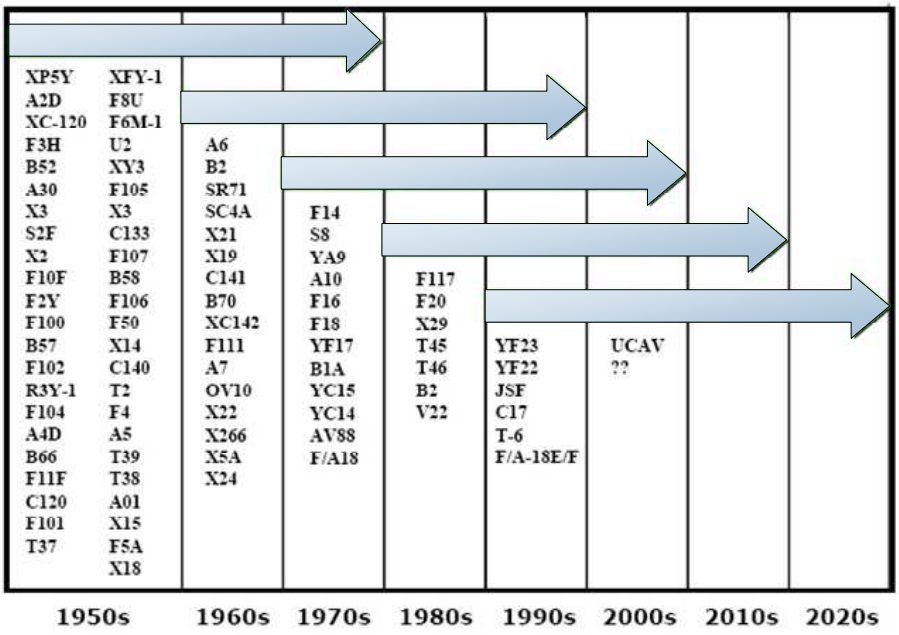


Figure 1.2: New US military project and typical career length of an engineer[3].

Facing these challenges, aerospace companies are looking for ways to keep an edge over the competition, by improving their processes. Since most process improvement techniques, such as lean principles, are widely available to most companies, the only way they can differentiate is by focussing on the knowledge available inside the company. Besides managing the knowledge already available, a knowledge creating company should be promoted.

The research presented in this thesis will contribute to answering the question of how to deal with the increasing complexity of the design problems aerospace companies are facing, and how to manage the knowledge inside a company. A knowledge-based approach for dealing with complex problems in the detailed design phase will be promoted, using the design of fibre metal laminate (FML) fuselage panels as a case study.

1.1 The challenge in dealing with complexity

The increasing complexity of aircraft results in further specialisation of disciplines. The challenge of complying with more and more requirements asks for a multidisciplinary approach, where the disciplines involved work in a concurrent way. How to efficiently come to an optimum solution to a design problem is the research field of Multidisciplinary Design Optimisation (MDO). However, successful representing a design problem in a mathematical format, which can be solved by optimisation algorithms, has been prevented by the high complexity of real world design problems[4]. Finding solutions to real world problems relies on capturing the way engineers decompose and solve problems.

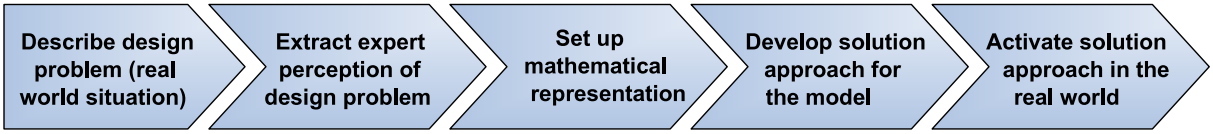


Figure 1.3: Method for the development of a solution to design problems.

Ackoff[5] discusses the importance of defining a problem, not just as a separate problem, but seen in its real-world where it is linked to other processes. The solution approach should not merely focus on finding the optimum for the isolated problem, but also on the efficiency of the solution within this real-world. Therefore, the first step is to describe the problem in the context of the multi-disciplinary environment it is situated in, see figure 1.3. Next step is to create an expert view of the problem, giving an explicit description of the problem in terms of system elements and their attributes, activities, relations, objectives and solution domain. The expert view has to be transformed into a well-defined mathematical model. Well-defined means that solutions can be expressed in a set of design variables, evaluated and given a score (using an objective function), and constraints define a feasible solutions space. Finally, a solution approach to the mathematical model is to be developed and implemented in the real world.

1.2 Objective

The challenges facing the aerospace industry can be summarised as follow: increasing complexity of systems, in combination with a decrease in availability of intellectual resources and reduced knowledge transfer opportunities. A case study encompassing the above challenges is the detailed design of FML fuselage panels. Objective for this research

is to develop and activate a knowledge based solution system for this design problem, reducing required intellectual resources, design process lead-time, and promoting the storing and reuse of engineering knowledge. A method for solving complex design problems will be presented, by formalising the approach for the development of the solution system (figure 1.4).

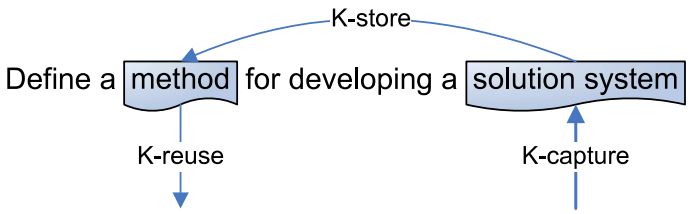


Figure 1.4: Thesis objective: capture and store problem solving knowledge, to be able to reuse it.

1.2.1 Developing a solution system to the FML design problem

The development of the solutions system starts with a formal description of the design problem. A formal model of a design problem in general consists of a process and a product view. The process view is needed to state the objective of the solution system, and to define requirements with respect to the way this objective is achieved. The product view is used to define the solution domain, and it contains requirements that limit the possible number of solutions. Both the process and product view are based on expert domain knowledge, to guarantee that the solution is effective in solving the real world problem. For the FML design problem, an expert view on the problem itself and how he solves the problem are defined. When captured in a formal way, the expert knowledge can be stored in a solution finding system, based on the principles of knowledge based engineering.

Questions to be answered:

- What are the challenges faced in the detailed design of products, and in specific of FML fuselage panels

- What are the steps in defining a solution system for the detailed design problem

1.2.2 Implementing the solution system in the real world

The efficiency and effectiveness of the knowledge based solution system to find solutions to the complex design problem has to be evaluated. Efficiency can be expressed in the time needed to generate results, effectiveness in the ability to achieve the goals set. Efficiency is evaluated by looking at the solution system's ability to reduce the complexity of the solution finding process, i.e., how fast can the system generate solutions. Effectiveness can be expressed in the ability of the system to find useful solutions to the problem, and in the possible lead-time reduction of the total design process. The evaluation of both efficiency and effectiveness should be done by activating the solution system in the real world situation. Practical issues when activating the system in the engineering world should be identified and dealt with.

Questions to be answered:

- What are critical issues that need to be addressed in order to successfully activate the knowledge based solution system in the real engineering world.
- How can these issues be dealt with.

1.2.3 Formalising a method for finding solutions to design problems

To be able to reuse the knowledge on how to setup an efficient and effective solution system, the approach presented should be formalised and stored in a method. The elements that compose the solution system and the relations between the elements should be discussed. Critical element in the solution system is the solution finding strategy, contained in a so-called problem solving method, which should be able to deal with the complexity of real world design problems.

Questions to be answered:

- Can a problem solving method be used to store and re-use problem solving knowledge.
- How to define a problem solving method for complex design problems, independent of the problem at hand.

1.3 Thesis outline

The different steps in the development of KBE applications, see figure 1.5, will be discussed in the thesis. Chapter 2 identifies the challenges faced in the detailed design process, and shows how KBE can be used to deal with these challenges. As case study, a complex design problem in the detailed design of FML fuselage panels is presented in chapter 3. An expert view on the design problem is discussed in terms of product entities, design process activities and knowledge on how the expert currently solves the problem. Chapter 4 shows how the expert knowledge on the design problem is formalised to define a well-defined mathematical model. By combining mathematical solution finding knowledge with the expert view on how to solve the problem, a solution finding algorithm is setup. The efficiency of the algorithm, in terms of reducing the complexity of the solution finding process, is assessed. Chapter 5 discusses the final steps in the development of a solution system, where the solution system should be packaged in a KBE application and activated in the real world situation, i.e., the design process. Effectiveness of the application, in terms of finding solutions and reducing engineering resources and/or process lead-time, will be discussed. Furthermore, chapter 5 presents how issues, involving the activation of the KBE application in the real engineering world, can be dealt with. In order to store and re-use the problem solving knowledge obtained in the research, a problem solving method for complex design problems is presented in chapter 6.

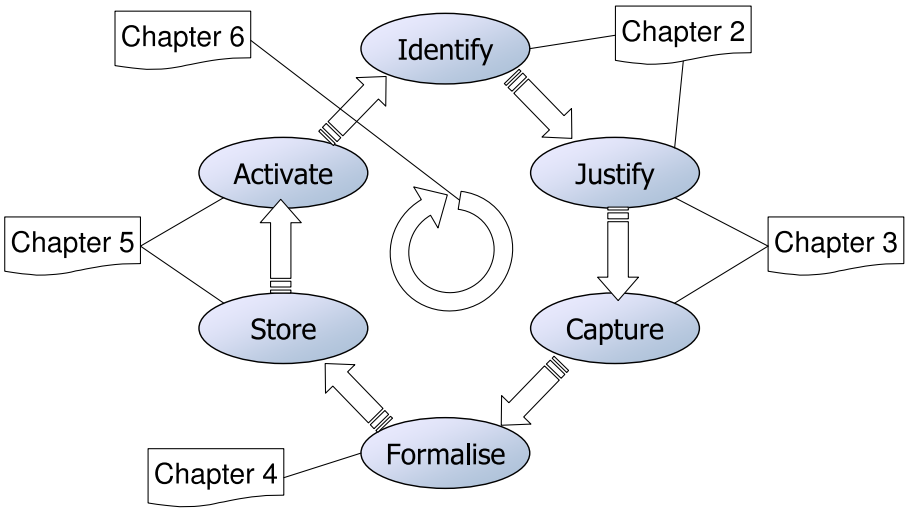


Figure 1.5: Lifecycle of the development of KBE applications [6].

Bibliography

[1] ACARE; European Aeronautics: A vision for 2020. *(http://acare4europe.com, white paper, 2001)*

[2] AIAA Technical Committee on MDO; Current State of the Art on Multidisciplinary Design Optimization. *(AIAA, ISBN 1-56347-021-7, 1991)*

[3] Hernandez C.: Intellectual Capital. *(The California Engineering Foundation, White Paper, 1999)*

[4] Alexandrov N.: Editorial-Multidisciplinary Design Optimization. *(Optimization and Engineering, Vol 6., pp. 5-7, 2005)*

[5] Ackoff R.L.: Optimization + objectivity = opt out. *(European Journal of Operational Research, Vol. 1, pp. 1-7, 1977)*

[6] Stokes M. (on behalf of the MOKA consortium): Managing Engineering Knowledge - MOKA: Methodology for Knowledge Based Engineering Applications. *(Professional Engineering Publishing Limited, Bury St Edmunds, UK, 2001)*

Chapter 2

Knowledge-based Solution for Detailed Design Problems

Design of aerospace systems is becoming more and more complex, because of the increasing number of requirements as shown in the previous chapter. In order to define solutions to these complex engineering problems, a process focussed approach is required. The flow of information between the different disciplines involved and between the various elements in the (sub)systems must be closely examined, to a priori identify and resolve issues in the process. The focus of this chapter will be on the detailed design process, and how the principles of knowledge based engineering (KBE) can help to improve the process flow.

First the challenges in the current design process are identified in section 2.1, followed by a justification of why KBE can help to deal with these challenges. This justification part consists of an explanation of the general principles of KBE (Section 2.2), of what the future design process will look like (2.3), and of a state-of-the-art of KBE (2.4). The need for a systematic approach for developing KBE applications will be discussed in section 2.5, followed by an overview of such an approach discussed in literature.

The following statements will be addressed:

- Implementing KBE will reduce the cost of design iterations by automating non-creative design process steps
- KBE turns knowledge into a company asset, by capturing, storing and re-using expert domain knowledge

To be able to correctly interpret the reasoning in this chapter, definitions of data, information and knowledge are given. Data consists of non-related facts, for instance

$\{a, b, 1, 3\}$. Information gives a meaning to data by specifying relations between data: $\{a + b = 1, b > 3\}$. Finally, knowledge is understanding information, the ability to see patterns in information and to react on that: if $a + b = 1$ and $b > 3$ then $a < -2$.

2.1 Information flow in the detailed design process

2.1.1 General design process

Before discussing the detailed design process, the design phases preceding will be briefly presented. A general approach for finding a solution to a design problem is to first explore different concept solutions, based on a list of requirements, see figure 2.1. This is called the conceptual design phase and has a diverging character. Following, during the preliminary design phase, more information on the performance of the concept solutions is obtained through multi-disciplinary analysis, resulting in a best solution by making a trade-off between the different concepts.

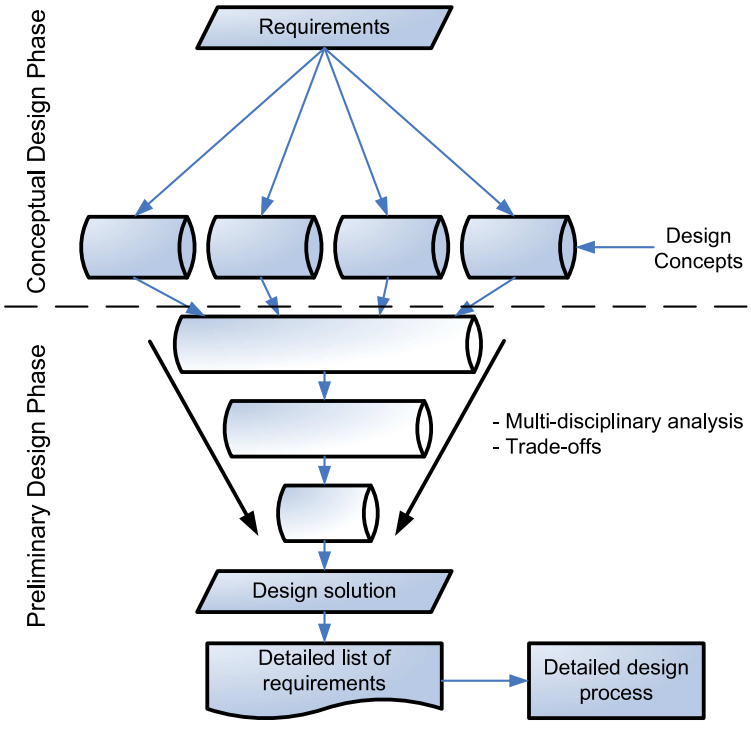


Figure 2.1: Diverging - converging character of the design process.[1]

The output of the preliminary design phase is used to setup a list of requirements for the detailed design process. Having defined a best solution, more detailed information needs to be obtained in order to be able to produce the design solution. This phase is called the

detailed design process. Figure 2.2 shows an schematic overview of the design process, as implemented traditionally at engineering companies.

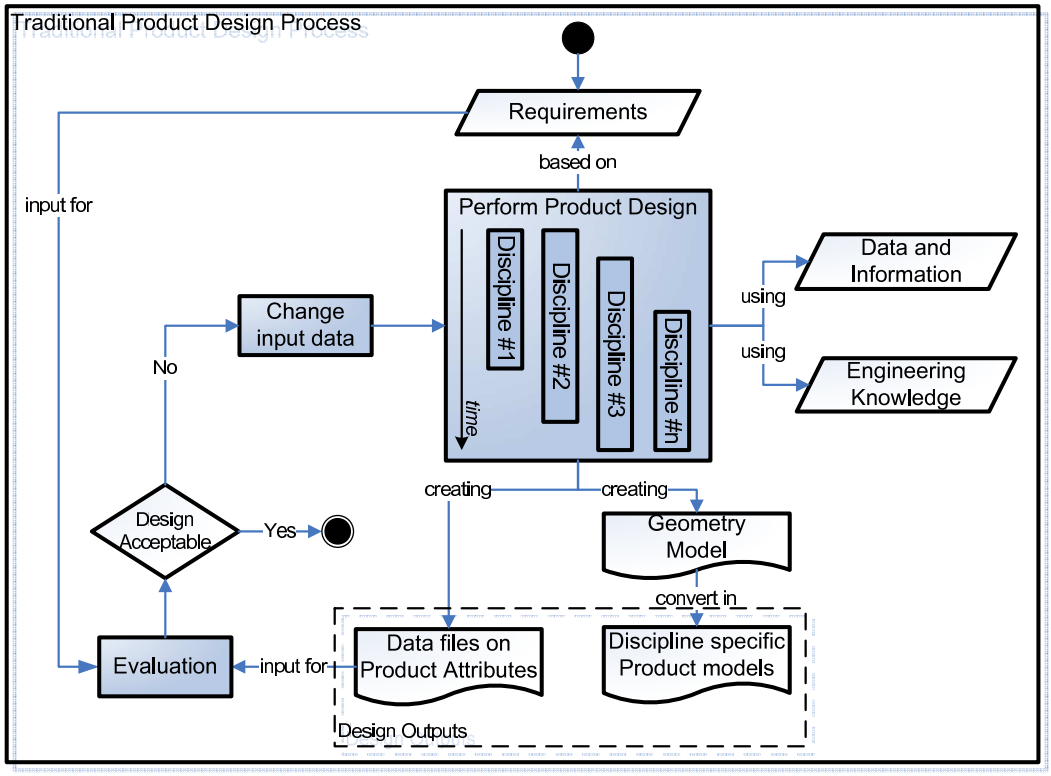


Figure 2.2: Traditional Product Design process.

The process starts with the data and information generated in the previous phases of the design process, i.e., conceptual and preliminary design. Based on the list of requirements and using knowledge on how to perform design tasks, different design outputs are generated. The design outputs consist of data files, describing the values of the product attributes, and discipline specific product models. The design outputs will have to be communicated between the different disciplines. In this transfer of information some challenges can be identified, since information stored in a product model of the submitting discipline cannot a priori be read by the receiving discipline. Different disciplines create different models for their analysis tasks. Often the information is manually transferred, in order to be useful as input for the next discipline. Furthermore, the generation of the design outputs is often a time consuming process, and is repeated during each cycle in the design process.

2.1.2 Relations in information

A further complicating factor in detailed design is the process of designing a system, consisting of different related system elements. A relation defines a link between the attributes of two entities or system elements. A change in attributes of one element can propagate a change in attributes of the other element. Figure 2.3 shows an N^2 matrix of different elements in a system and four possible ways of change propagation inside the system.

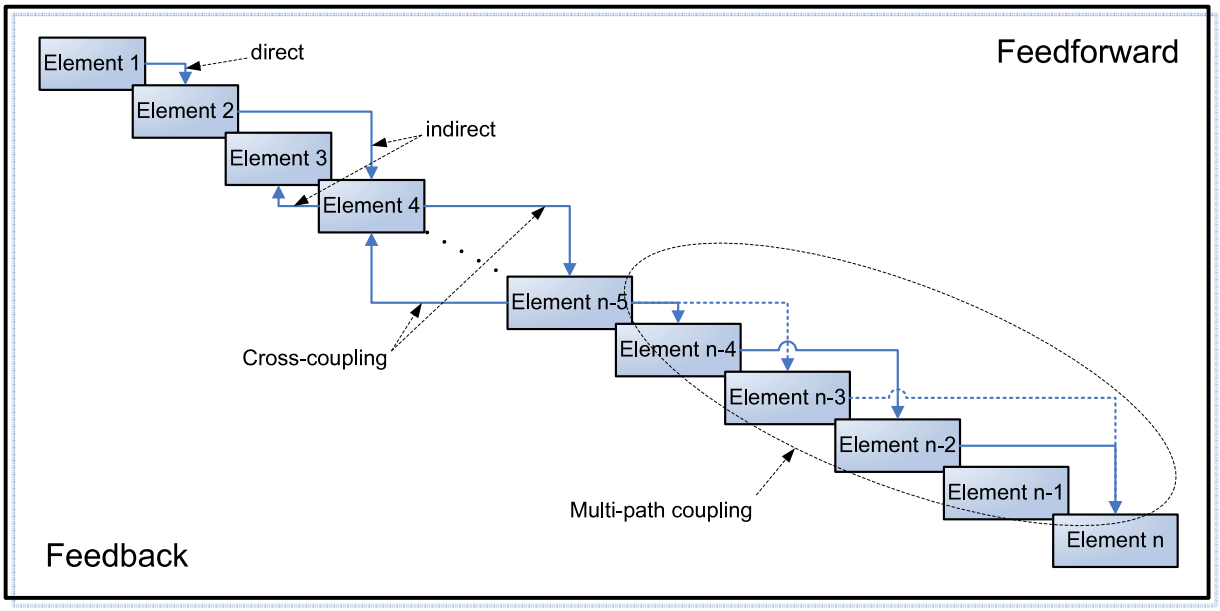


Figure 2.3: N^2 matrix of a system.

Propagation can be direct, where a change in element 1 causes a change in element 2, or indirect, where a change in element 2 cause a change in element 4 that impacts element 3. The indirect propagation in this example is cross-coupled if element 3 in figure 2.3 also impacts element 2. Finally, a relation can result in a multi-path coupling, if a change triggers two or more separate series of changes, intersecting again at a certain element.

Relations can be *unidirectional*, an attribute of element 1 impacts one attribute of element 2, *bidirectional*, an attribute of element 1 impacts one attribute of element 2 and viceversa, or *multidirectional*, attributes of element 1 impact multiple attributes of element 2 and viceversa. Note that bi- or multidirectional relations between elements constitute a cross-coupled change propagation. A different relation generalisation is by the nature of the change propagated, and Riviere[2] identified the following four relation types:

I **Formal**. A change in a system element requires a renewed release of the element and of the system it is part of.

II **Functional**. System elements are functionally related if a change in element 1 requires an engineer to assess if the related element can still fulfill its function, or needs to be redefined. For instance if the thickness of the skin is reduced, an engineer needs to assess if the selected rivet type can still fulfill the requirements.

III **Feature based**. A change in element 1 can directly induce a change in element 2 based on a feature relation. For instance changing a splice location requires the joggles in the back-up structure to change accordingly.

IV **Derivative**. A representation of element 1 needs to be updated if element 1 is changed. For instance the design outputs need to be updated if the product attributes are changed.

2.1.3 Iterative generation of information

The four types of relations are graphically represented in figure 2.4, showing the design process of a system consisting of three elements. Element three is a stringer, which is joined to the skin (element 2) by means of rivets (element 1). When changing for instance the thickness of the skin, an engineer will have to assess if the rivets still comply with the requirements. This relation is bidirectional, since a change in rivets because of for instance limited installation space, will require a re-evaluation of the skin. The information created for the skin serves as input information for the stringer 3 via a feature based relation. Shifting a thickness step in the skin will result in a change in the stringer. An issue arising is the format in which output information from one process can be transferred to the next, resulting in additional work for the engineers. A concurrent process flow is often selected, in order to reduce lead-time and identify conflicting requirements in an early stage of the design. However, since the detailed design process of element 2 has not been completed, the information that is used for the design of element 3 is not final. This will result in rework to implement the changes.

One of the attributes of efficient engineering is to minimise waste, in engineering terms reduce the amount of rework. This can be facilitated by providing the right input at the right time and right quantity. In order to ensure that requirements from the different disciplines are identified and implemented in an early stage, a concurrent process is

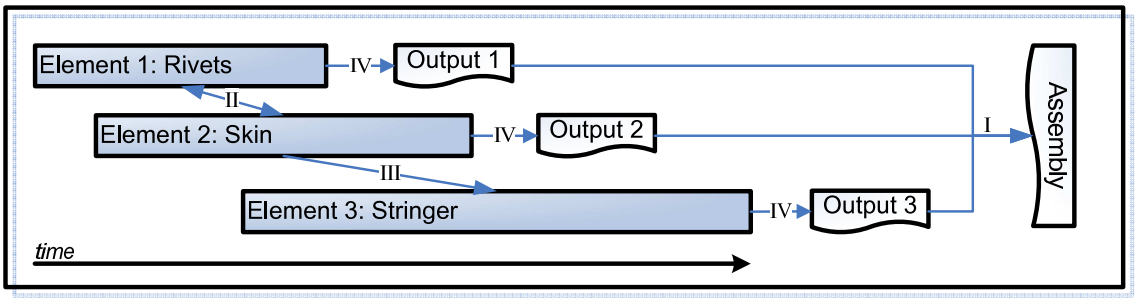


Figure 2.4: Exemplary design process of a stiffened skin system without KBE.

implemented. Expensive rework at the very end of the process can be prevented this way. However, in some cases the information used by an engineer to execute a task is not yet final, given the iterative character of the engineering process. In the case of a unidirectional relation between two system elements, the right input at the right time would require a more sequential process, where a final set of information is passed on between different engineering groups. In a concurrent process, one should always clearly differentiate between simultaneous involvement and simultaneous activities.

The iterative character of a design process results in a repetition of various tasks. These repetitive tasks, once a good practice has been established, need little to no creativity from the engineer. Automating these tasks reduces the cost of iterations and the time required for the design process, freeing more time for the creative part of the design process.

2.2 Principles of knowledge based engineering

Knowledge based engineering is the science of identifying, capturing, storing and re-using expert knowledge. The objective of KBE is to automate engineering process steps, once the knowledge needed to execute the step is captured and stored in a consistent format. Expert knowledge is a combination of both problem specific information and knowledge, for instance best practices or design rules. According to Luger[3], expert knowledge can be defined as a combination of a theoretical understanding of the problem and a collection of heuristic problem-solving rules that experience has shown to be effective in the domain.

The origin of KBE is in the field of knowledge based systems (KBS) or expert systems (ES). KBE can be seen as a sub-set of expert systems, focussing on implementation in the field of engineering. General philosophy for generating expert systems is to establish rules of inference, offering public paths for drawing intelligent conclusions from existing knowledge[4]. Expert systems are designed to find solutions to complex problems by

combining formalised expert knowledge, stored in a knowledge base, with an 'inference engine' to find solutions to the problem.

KBE systems differ from expert systems because they combine engineering knowledge with geometrical capabilities, needed in the engineering environment. The KBE application is aiming at storing product and process knowledge, in order to model the design process and automate the different tasks within the process. In an object-oriented environment, objects can be instantiated using a knowledge base and input data specified by a user, forming a model of the product. The product model represents the engineering intent behind the design, containing a high level of information and knowledge on both process and product, see figure 2.5.

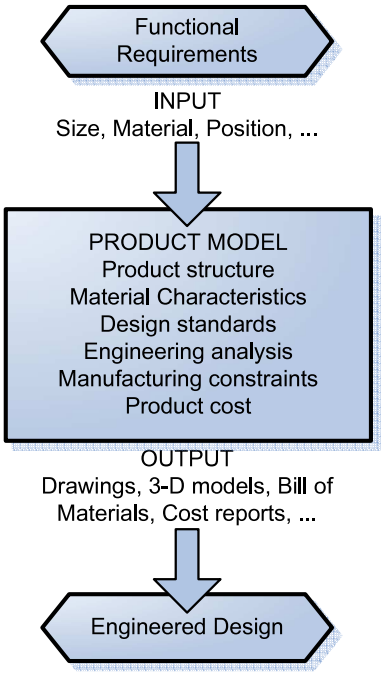


Figure 2.5: The product model containing process and product information[5].

The product model can contain for instance physical information on what materials are used, geometry of the different parts, and knowledge relating to how the different discipline tasks are executed. From the product model for instance, various models can automatically be derived, containing information for a specific process step, such as FE modeling, detailed design, and production. The information stored in the product model can be used to evaluate the performance of the product, and compare it to the requirements. An external inference engine or the user then alters the input data until the product performance is according to requirements.

2.3 The design process using KBE

This section discusses how the implementation of KBE effects the design process. Since information and knowledge on the design process and product attributes are stored in the product model, the process of performing the product design (see figure 2.2) and the creation of the design output can be automated. Figure 2.6 shows a schematic representation of the design process using KBE.

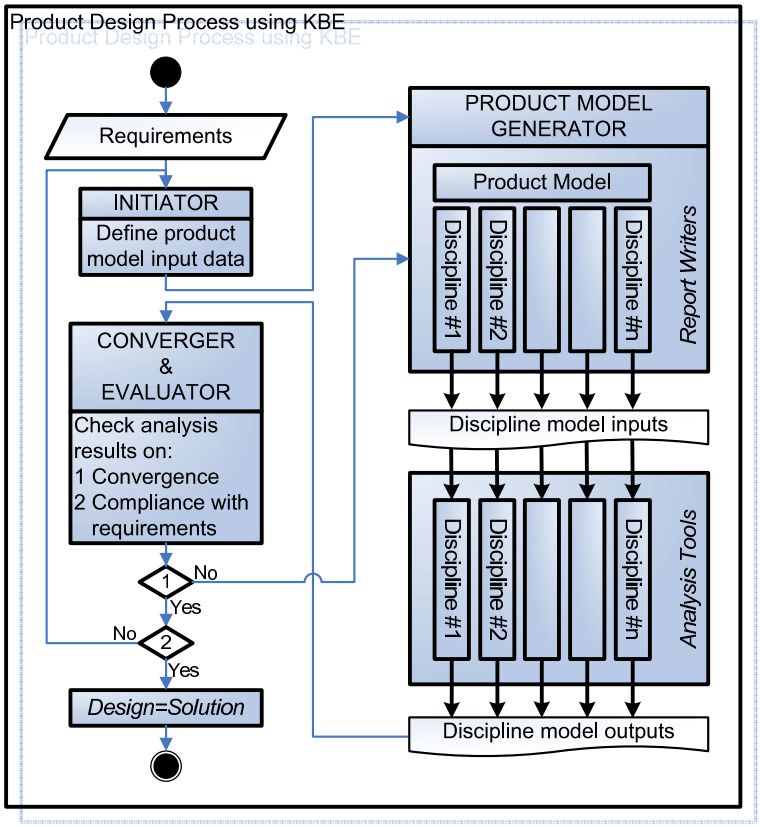


Figure 2.6: Design process using the principles of knowledge based engineering[6].

Using input data specified by an initiator (either a user or software application) and the expert knowledge base, the product model can be instantiated. Using the process knowledge, different report files are created, which serve as input for the discipline specific analysis tools. The needed design outputs, such as drawings, 3-D models, bill of materials, cost reports, are generated automatically. Finally the performance of the product has to be compared to the initially stated list of requirements, in order to assess if the design is a solution to the design problem.

A high level of process automation in the detailed design can reduce lead-time considerably. All disciplines are working concurrently using the same product model, eliminating

the issues of transforming information from one discipline to the next. Using the formal expert knowledge, every discipline can extract its own set of information from the product, perform an analysis and export its design outputs if needed.

Implementing the principles of KBE in the process of designing a system, more benefit can be achieved. As discussed in the example system design process in section 2.1, the output information of element 2 is needed as input information for element 3. By making use of generative product models, a consistent way of sharing information can be achieved, reducing the amount of work needed to transfer the information from one process to the next. Furthermore, the cost of iterations in the design process of the individual elements is reduced by automation, and a reduction in lead-time can be achieved, see figure 2.7. This figure shows a schematic representation of the design process of a system of elements when implementing KBE. Because of the reduction in lead-time of the design process for element 2, the start of the design process for element 3 can be postponed until all information needed is available. This statement is only valid if the relation between element 2 and 3 is of a unidirectional type. The lead-time for element 3 will not only be reduced by a high level of automation, but also because of a reduction in design cycles, which were required as a result of incorrect input information in the traditional system design process. A more sequential process will be the result, where a complete set of information is transferred from one process to the next.

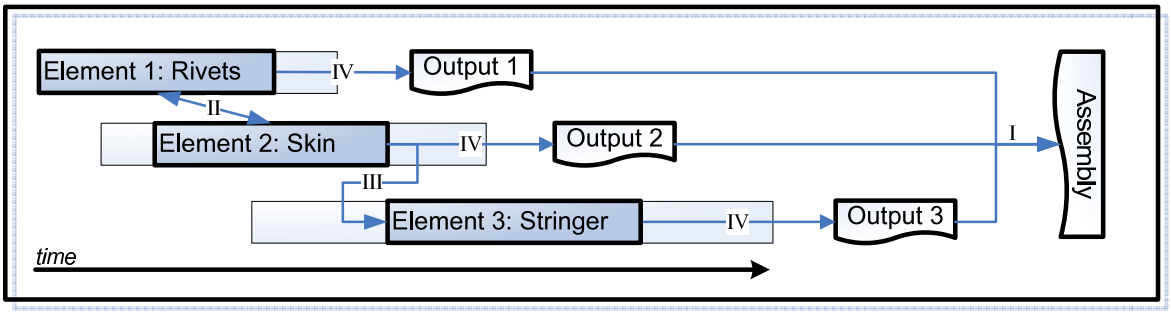


Figure 2.7: Exemplary design process of a stiffened skin system using KBE.

Concluding, the implementation of KBE in the design process will have the following implications on the issues stated in section 2.1:

1. Formalising knowledge acquired during other projects to be able to store it for re-use during future projects.

2. Reduction of the cost of iterations in the design process, making it possible to quickly incorporate input changes and resulting in a decrease in lead-time.

3. Reduction in resources by postponing engineering tasks until a complete set of information is available.

2.4 State-of-the-art

KBE applications are being developed in an increasing scale, especially in large companies in the automotive and aerospace industry. The first KBE development software became available in the 1980s, and steadily the number of applications is increasing. Main reasons for developing KBE applications are reduction in engineering time and cost, and improvements in performance and quality of the products[5]. Main fields of KBE application are the detailed design and manufacturing[7]. KBE applications can also facilitate the conceptual design phase, by moving detailed knowledge from the end to the start of the design process. This approach is especially important because the cost of a product is to a large extend determined in the conceptual design phase, given its irreversible impact on the detailed design and production phase[8]. Table 2.1 shows some examples of reductions in lead-time which have been achieved for different product ranges.

Table 2.1: Industry examples of KBE applications and achieved lead-time reduction

Application	**Lead-time reduction / Improved concept exploration**	
Conceptual aircraft design	60 more concepts	[5]
Windscreen wiper system	weeks → min	[5]
Jaguar bonnet design	8 weeks → 20 min	[9]
Wingbox redesign	8000 hrs → 10 hrs	[9]
Airfoil shape optimisation	2 months → 4 days	[10]
Compressor design	10 days → 1 day	[10]

Besides aircraft integrators, also suppliers could benefit from KBE applications, because they supply similar components to different companies. Nevertheless, KBE hasn't been implemented as widespread as for instance traditional CAD systems. KBE application have been developed as pilots, proving the potential for the business on the short run. Often the large investments in development and expensive resources has scared management from implementing KBE as a company design standard. Nevertheless, the larger companies envision future KBE developments to result in integrated product design applications[11].

2.5 Methodology for developing KBE applications

KBE applications are often developed on an ad hoc basis, as soon as a need is identified. The application is the main deliverable, and an immediate return on investment should be the result. This approach disregards the need for thorough capturing and documenting of the product and process knowledge. Indeed, a formalised approach of capturing, storing and re-using knowledge will increase the efficiency of the KBE development since the knowledge is in a consistent format, understandable for every discipline involved. Furthermore, knowledge loss because of incorrect modeling of the knowledge or misinterpretation can be prevented. A consistent format will also make maintenance of the application more transparent. Finally re-use of the knowledge is facilitated, without having to go through the code of the application to retrieve the knowledge.

A methodology focussing on the development of KBE applications is MOKA: 'Methodology and software tools Oriented to Knowledge based engineering Applications'[9]. The definition of KBE, shown in section 2.2 already contains the four general phases in the development of a KBE application: Identify, Capture, Store and Re-use. Figure 2.8 shows the different phases in the development of a KBE application as defined in MOKA.

The **Identify** phase is intended to scout opportunities for KBE development or maintenance, and to evaluate if the investments can be justified. The actors in this phase are the program manager and the engineers involved in the design process, who are in general the intended users. The program manager is able to make a decision on whether to proceed and how to implement the application in the process. The users specify requirements on functionalities the application should have. Next phase is the **Capture** phase, where the knowledge required for the application is captured and formalised in a knowledge repository. This phase is crucial in assuring that knowledge becomes an asset of the company, because it is expressed in an explicit and consistent manner. The actors in this phase are the domain expert, providing the required knowledge, and the knowledge engineer, responsible for extracting and formalising the knowledge. Having incorporated the knowledge in a repository, the next phase is to **Store** the knowledge in the actual KBE application. Two actors are involved in this phase, i.e., the knowledge engineer and the developer. Here the multi-disciplinary role of the knowledge engineer becomes evident. Besides an engineering background to be able to communicate with the domain expert, the knowledge engineer should also be able to formalise the knowledge in such a way that it can be coded by the developer. Finally in the **Re-use** phase, the KBE system is implemented in the design process. The actors here are again the program manager,

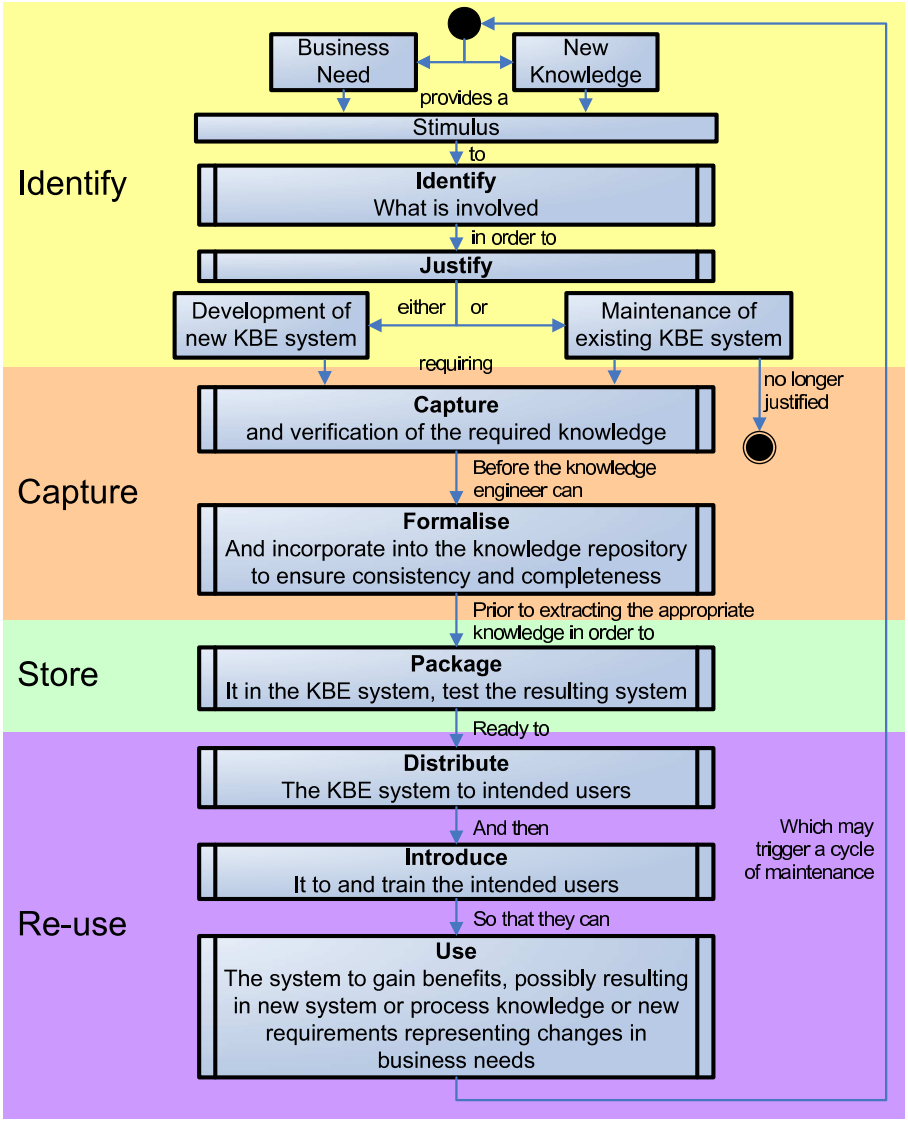


Figure 2.8: The KBE development lifecycle according to MOKA[12].

responsible for implementing the application as efficient as possible in the design process, and the user operating the system.

2.6 Conclusions

The design process of aircraft is becoming more and more complex because of the increasing amount of design requirements. Implementing all requirements demands for control on the flow of information between different disciplines and between different elements in aircraft (sub)systems. Information transfer of a submitting discipline to the receiving is often complicated, because different product models are used for analysis, requiring a

redefinition of the information. Furthermore, the time consuming process of generating design outputs has to be repeated at each design cycle.

Knowledge based engineering (KBE) is the science of identifying, capturing, storing and re-using expert knowledge. KBE proposes that engineering steps can be automated once the knowledge needed to execute the step is captured and stored in a consistent format. A generative product model is defined, based on input information and expert domain knowledge. The different disciplines involved receive a set of information needed for their analysis from the product model, making the redefinition of information superfluous. The cost of iterations in the process is reduced by automation, making it possible to quickly incorporate input changes and decreasing the lead-time. In addition to lead-time reduction, implementing a KBE application in an efficient way will also result a reduction in resources by postponing engineering tasks until a complete set of input information is available.

Bibliography

[1] Tooren M.J.L. van: Sustainable Knowledge Growth, Inaugural speech. (*Delft, Delft University of Technology, 2003*)

[2] Riviere A.: Aircraft Component Impact Analysis: State of the Art. *(Vivace, 6th Framework Project, www.vivaceproject.com, 2006)*

[3] Luger G.F.: Artificial Intelligence: Structures and Strategies for Complex Problem Solving. *(Addison-Wesley, 5th edition, 2005)*

[4] Rhem A.J.: UML for Developing Knowledge Management Systems. *(Auerbach Publications, 2006)*

[5] Cooper S., Fan I., Li G.: Achieving competitive advantage through Knowledge-Based Engineering -A best practice guide. *(White paper prepared for the Department of Trade and Industry, University Cranfield, U.K., 2001)*

[6] Tooren M.J.L. van, Schut E.J., Berends J.P.T.J.: Design Feasilisation using Knowledge Based Engineering and Optimisation Techniques. *(44th AIAA Aerospace Sciences Meeting and Exhibit, Reno, USA, 2006)*

[7] Haas R., Sinha M.: Concurrent engineering at Airbus - a case study. *(Int. journal of manufacturing technology and Management, Vol. 6, Nos. 3/4, 2004)*

[8] Chapman C.B., Pinfold M.: Design engineering to rethink the solution using knowledge based engineering. *(Knowledge-Based Systems, Vol 12, p257-267, 1999)*

[9] Stokes M. (on behalf of the MOKA consortium): Managing Engineering Knowledge - MOKA: Methodology for Knowledge Based Engineering Applications. *(Professional Engineering Publishing Limited, Bury St Edmunds, UK, 2001)*

[10] Information obtained from the Engineous website. *(www.engineous.com, october 2006)*

[11] Mohaghegh M.: Evolution of Structures Design Philosophy and Criteria. *(45th AIAA Structures, Structural Dynamics & Materials Conference, Palm Springs, USA, 2004)*

[12] Oldham K., Kneebone S., Callot M.: Moka - A methodology and tools Oriented to Knowledge-based engineering applications. *(Advances in Design and Manufacturing, Vol. 8, Proceedings of the Conference on Integration in Manufacturing, Göteborg, Sweden, 1998)*

Chapter 3

Complexity in Detailed Design of Fibre Metal Laminate Structures

The detailed design of aerospace structures is executed in a multidisciplinary environment. Each department has its own set of requirements, for instance with respect to airworthiness or producibility. During the detailed design process, a product definition must be created fulfilling these requirements. Given the high complexity of aerospace structures, these requirements can become conflicting. Resolving these conflicts requires an iterative solution finding approach, in combination with a large knowledge of the material specific design and manufacturing principles.

This chapter will discuss a design problem in the detailed design of Fibre Metal Laminate (FML) structures. First an introduction to the FML structure built-up and production process is presented in section 3.1. Next an overview of the real world in which the problem is situated will be given. This is done in section 3.2 by describing the different design process steps and the flow of information between these steps. To better understand the challenges faced during the detailed design, section 3.3 presents a case study of a double curved FML panel. Following, an expert view on the design problem will be presented in section 3.4, needed to be able to define a solution approach to the problem.

The following statements will be addressed:

- Creating a feasible FML product model is an iterative process in itself.
- Unified Modeling Language (UML) can be used to formalise expert knowledge in the engineering domain.

3.1 Introduction to FML fuselage structures

Fibre Metal Laminates (FML) are a family of hybrid materials, developed for aerospace applications. Today FML are applied in large upper sections of the front and aft fuselage of the Airbus A380. Strong point of these laminates are the improved damage tolerance characteristics compared to monolithic aluminium. FML consist of alternating metal and fibre reinforced polymer layers, placed in a single or double curved mould and bonded together by curing in an autoclave. The thin flat aluminium sheets are placed in the mould without preforming, called the free forming technique[1]. Placing such flat metal sheet in a double curved mould, can result in unallowable wrinkling of the sheet in combination with springback (figure 3.1). Decreasing the width of the sheet will eliminate sheet wrinkling and reduce the amount of springback. However, to reduce part count in a fuselage structure, the dimensions of the individual panels should be increased. To achieve larger panels, the aluminium sheets are joined in so-called splice areas using adhesive film, see figure 3.1.

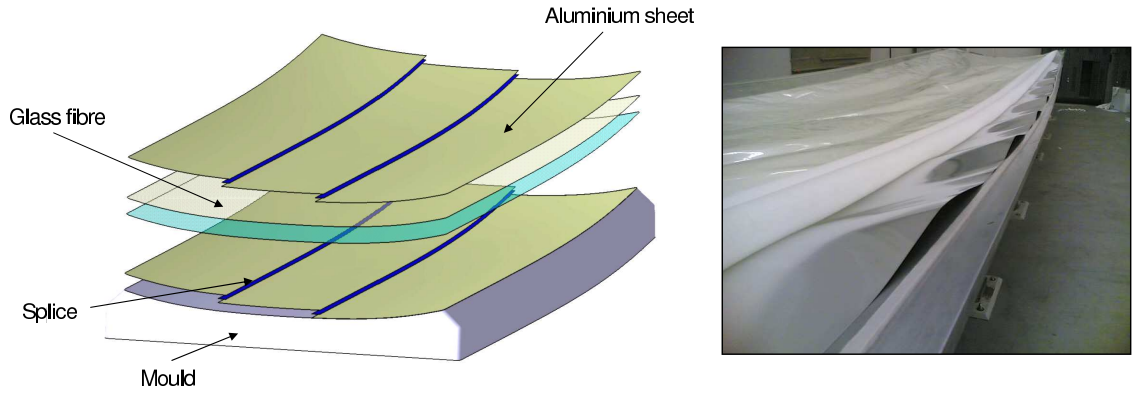


Figure 3.1: Increasing the width of FML panels by splicing the metal sheets.

A typical fuselage structure consists of a relative thin skin, which is joined to a backup structure consisting of frames and stringers, see figure 3.2. The joining techniques are either bonding or riveting for the stringers and riveting for the frames. The fuselage structure is divided in several panels, which are joined in circumferential direction by means of lap-joints and in longitudinal direction using butt-joints.

From an engineering point of view, FML structures are like composite structures, where besides the structure also the material (the laminate) itself has to be designed. Given the fact that FML contain various layers with a specific material type and/or orientation, the laminate properties can be tailored locally to achieve an optimum structure.

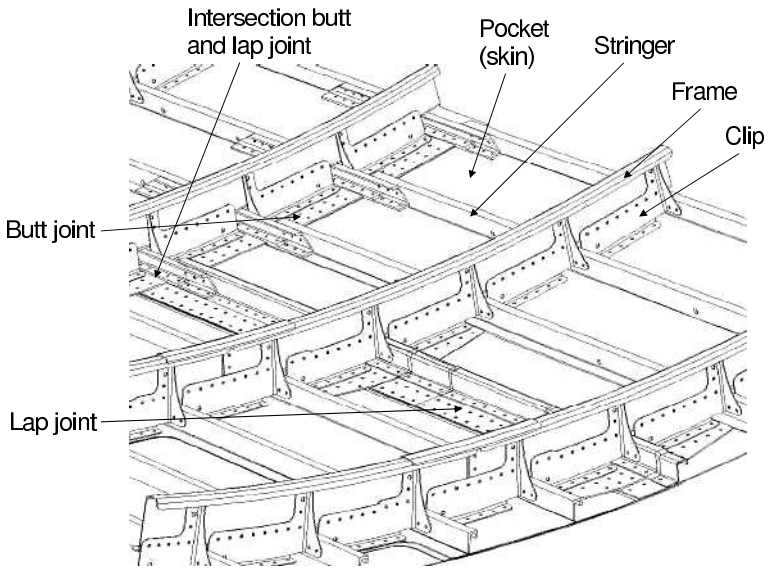


Figure 3.2: Description of a general fuselage structure.

3.2 Current FML detailed design process

The current detailed design process at Stork Fokker Aerospace (FAESP) for Glare panels is schematically represented in figure 3.3 as a standard input-process-output model. The required input consists of the output of the preliminary design phase, and of documents containing engineering requirements. The output of the process are documents and computer models, containing sufficient information for the production department to be able to prepare the production definition dossier, which is needed at the production site. The three parts in the process model will be discussed in more detail in this section.

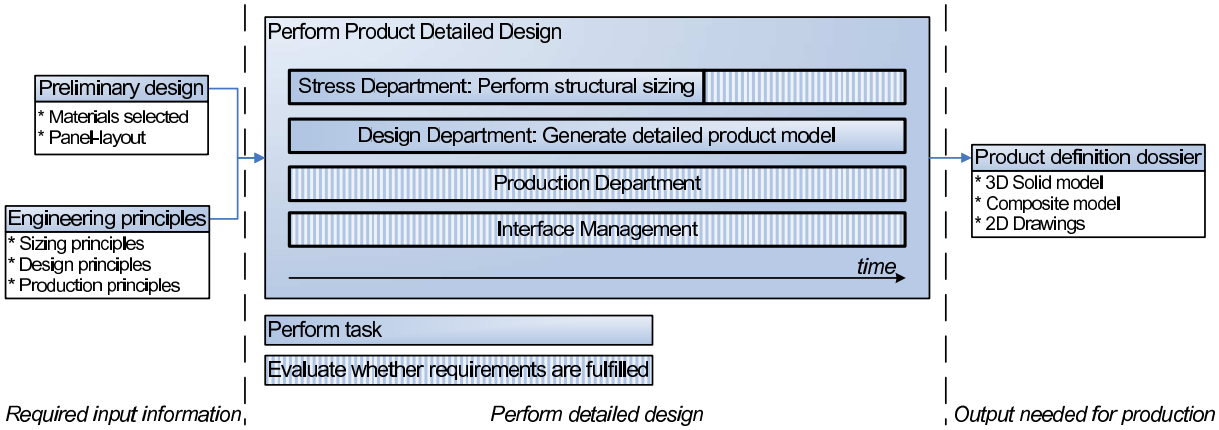


Figure 3.3: Flowchart of the current FML engineering process at FAESP.

3.2.1 Required input information

The required input information consists of the output of the preceding preliminary design, and requirements gathered from all disciplines involved. During the preliminary design of the fuselage, the positions and dimensions of the longitudinal and circumferential joints of the fuselage structure are determined, dividing it into individual panels. Furthermore, the material for each structural component is determined in a global optimisation of the fuselage.

The discipline specific requirements are summarised in the sizing, design and manufacturing principles. These principles contain guidelines for the detailed design phase, which are based on hard requirements ensuring airworthiness, producibility and product quality. A deviation from a guideline is allowed on a case to case basis, and only if the underlying hard requirement is not violated. The guidelines are obtained through testing or based on good practice / experience. The sizing principles describe, for instance, the material allowables that are needed for the structural sizing of the panels. The design principles contain information on structural details such as longitudinal and circumferential panel joints. Furthermore, the design principles discuss additional Glare specific guidelines, for instance shown in figure 3.4, indicating the minimum dimensions of the aluminium sheet overlaps in a splice area. The manufacturing principles describe requirements to ensure producibility. An example is a limitation on the width of the aluminium sheets in the laminate, because of sheet-wrinkling and spring-back from the mold in the case of double curved shapes.

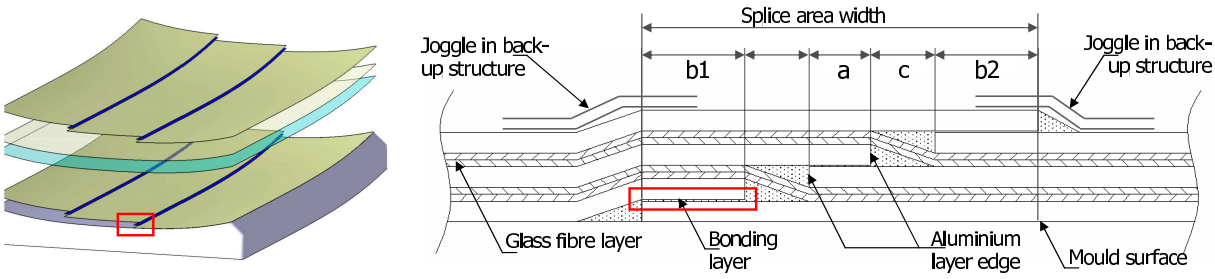


Figure 3.4: Design requirements on a typical splice area. a, b1, b2 and c are minimum dimensions

3.2.2 Perform detailed product design

Figure 3.5 shows an activity diagram of the product detailed design process. The first step in the process is the sizing of the structural components, executed by the stress department (see figure 3.3). For this purpose a FE model of the fuselage is made, and using an optimisation procedure, the dimensions of the structure resulting in minimum weight are determined. The following detailed information is determined:

- Minimum required skin thickness in terms of number of metal and fibre reinforced polymer layers
- Frame and stringer types
- Rivet types and pitches
- Minimum number of rivets in the frame-skin joint

The output of the structural sizing is called the theoretical panel design, summarising the above stated information.

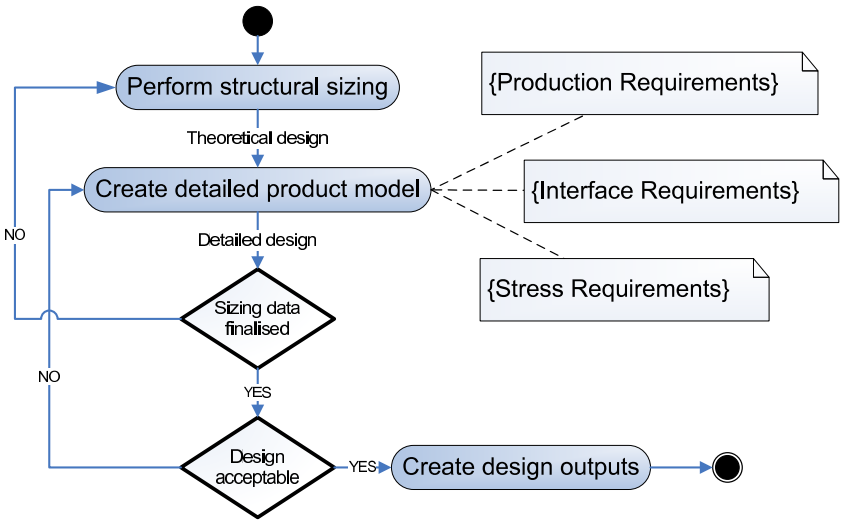


Figure 3.5: Activity diagram of the detailed product design process.

Next step in the process is to create the detailed product model, implementing the requirements from the disciplines involved. The production department has requirements with respect to producibility, the interface management group ensures that the different panels can be joined according to requirements. Creating the detailed product model step is done by the design department. During this step the laminate built-up is defined,

resulting in a detailed definition of the dimensions, location and stacking sequence of the laminate constituents. The 'create detailed product model' step can be split in the following consecutive steps, see figure 3.6:

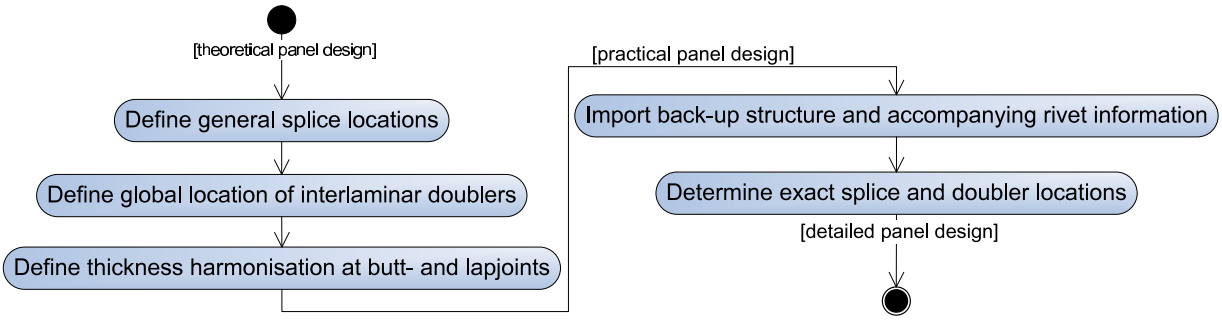


Figure 3.6: Activity diagram of the 'create detailed product model' process step in figure 3.5.

The first four steps result in a practical panel design, where general splice and doubler locations are determined, according to the stress, production and interface requirements. The theoretical design, generated by the stress department, is used as input. It is up to the designers skill to convert this theoretical design into a design that can actually be produced. Next the practical design has to be converted into a detailed design, defining the exact splice and doubler dimensions and locations. The dimensions are restricted by the design principles, like for instance requirements on the riveting of the skin to the back-up structure. Section 3.3 will give some examples of these requirements.

As indicated in figure 3.3, the stress and design department work concurrently, together with the production and interface departments. The iterative character of the design process results in changes in the sizing data, requiring a loop in the detailed product design process. Finally, all departments involved check the detailed product model in order to evaluate if their requirements are fulfilled. If so, then the design outputs needed for the production department can be generated.

3.2.3 Output information needed for production

The output information is stored in the product definition dossier. This dossier contains documents and computer models of the product, and contains all information needed by the production department to be able to generate the production definition dossier. The product definition dossier consists of:

- 3D solid model for digital mock-up and machining

- A composite model, containing a definition of all the layers in the laminate. Information in this model is used for cutting the layers from the raw material and directing the Laser Projection System (LPS)

- 2D laminate cross-sections to get a good insight in the built-up of the laminate. These cross-sections will be used by the departments to evaluate if the design is acceptable, and for the definition of future repairs.

3.3 Design Case Study

To get more insight in the complexity of the detailed design problem, a design case study of a double curved fuselage panel is presented (figure 3.7). Different types of requirements are discussed, and by means of the case study it is demonstrated how they can become conflicting.

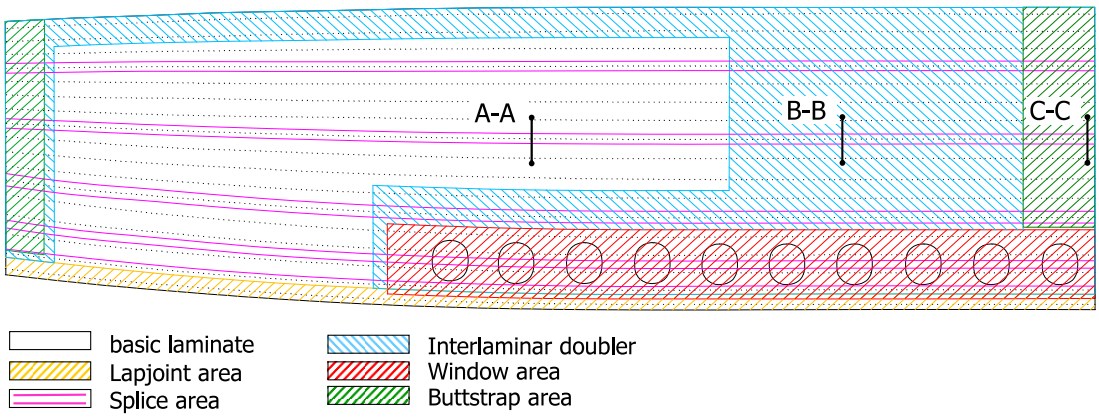


Figure 3.7: Practical layout of the panel.

The laminate consists of a basic laminate and interlaminar doublers to locally increase the thickness. Interlaminar indicates that the additional layers are located inside the laminate, see figure 3.10. Because of the fact that the panel is joined in longitudinal direction by lap joints and in circumferential direction by butt joints, thickness steps are not preferred. The minimum thickness of the panel in these joining areas is defined by the interface requirements. If needed, layers are added at the butt- and lapjoint areas. Limitations on the width of the aluminium sheets result in a minimum of four longitudinal splices required, shown in figure 3.7.

First a practical design of the panel is made, showing general splice and doubler dimensions. Next the back-up structure and accompanying joint information has to be imported, determining exact dimensions of splices and doublers. To get a good understanding of the laminate built-up and to be able to determine the exact dimensions of splices and doublers, cross-sections are made at every frame station. Combining these cross-sections with the topview of the panel, a kind of 2.5D image of the panel is created. When changing the laminate lay-up at one cross-section, it should be noted that the cross-sections at other locations in the panel will also change. For instance when changing the location of a splice area at the start of the panel, the splice area will change throughout the panel, influencing all related cross-sections.

3.3.1 Splice design at the three cross-sections

Three cross-sections at different frame stations are made (figure 3.7) for this case study. The topview of the panel in figure 3.7 shows that an interlaminar doubler is inserted to increase the thickness of the laminate, going from A-A to B-B. Furthermore, additional doublers are required to increase the thickness of the laminate towards the butt strap areas at C-C. Since the thickness of the laminate is increased at the location of a splice area, see figure 3.4, one of these doublers has to be split, ensuring a constant thickness distribution along the edge in circumferential direction (flattening doubler in figure 3.11). The stringer locations are fixed, the dimensions and locations of the doublers and the splices are to be shifted until the requirements with respect to the back-up structure are met.

Many requirements described in the design principles refer to the riveting of the skin to the back-up structure. For a splice area they can be summarised as follows:

- In general the distance between a rivet and a free aluminium edge in the laminate should be no less than the edge distance **c**, depending on the rivet diameter.
- The allowable rivet pitch is limited by a minimum d_{min} and maximum d_{max} value also relative to the rivet diameter.
- Rivets should be placed at a certain distance **a** from joggle edges, see figure 3.8.

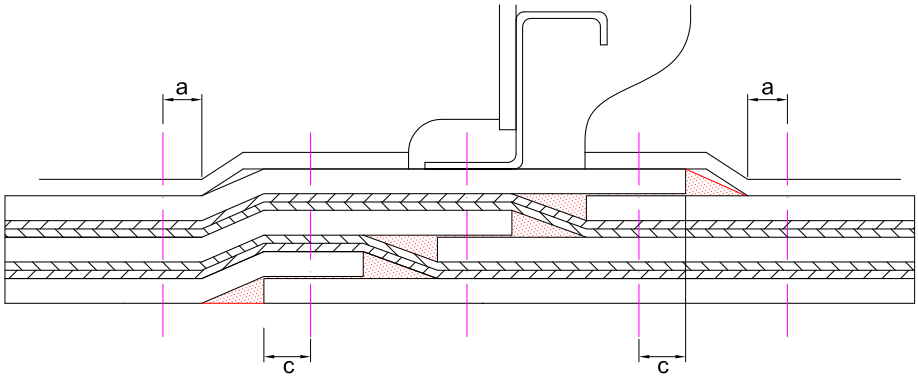


Figure 3.8: Requirements on the rivet positions.

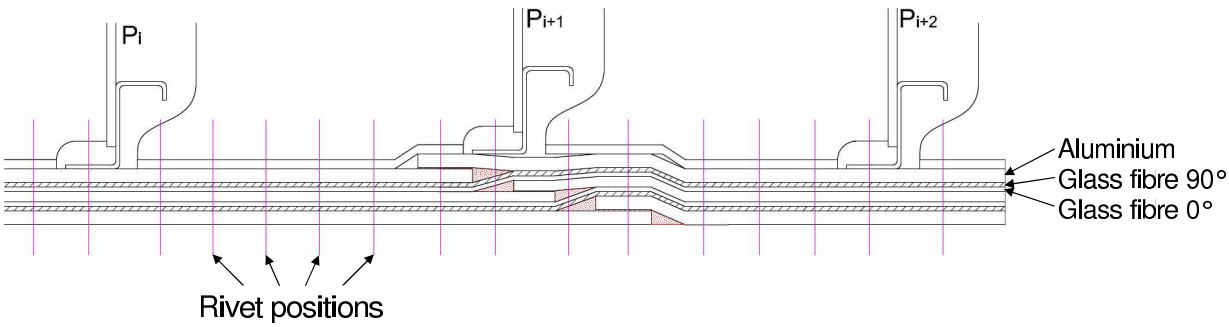


Figure 3.9: Regular splice geometry at A-A.

At cross-section A-A a splice is located at a stringer, called P_{i+1}, see figure 3.9.

The requirements with respect to the riveting are fulfilled for these splice dimensions. However, the riveting requirements at B-B and C-C should also be evaluated. For this reason two additional cross-sections have been made, section B-B (figure 3.10) and section C-C (figure 3.11). Cross-section B-B shows how the interlaminar doubler is added inside the laminate and is being spliced in de splice area. Cross-section C-C shows the additional flattening doublers for ensuring a flat circumferential joint area.

Both figures show that not all design requirements with respect to the riveting of the back-up structure can be fulfilled. In cross-section B-B the rivet pitch between the rivets marked with an 'x' is too large. The splice layout in cross-section C-C indicates an edge distance 'y' which is too small. In order to comply with the riveting requirements, the dimensions and location of the splice area and interlaminar doublers should be altered. However, altering the dimensions of the splice area at section B-B influences the splice dimensions at the other cross-sections, resulting in possible requirement violations at these cross-sections.

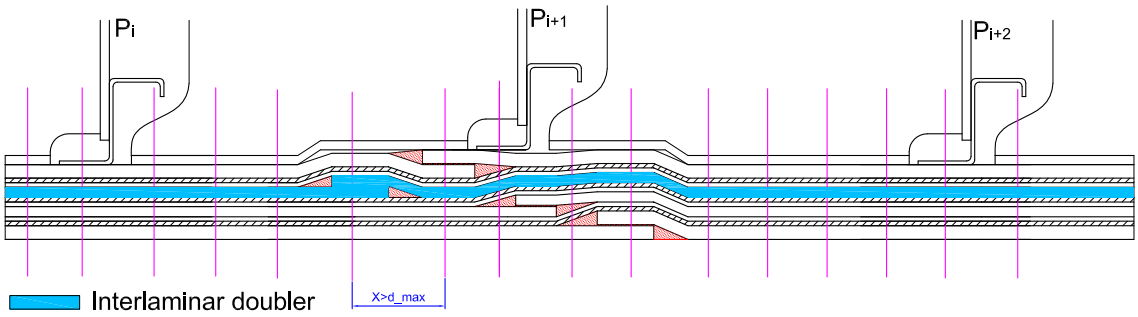


Figure 3.10: Laminate cross-section at B-B.

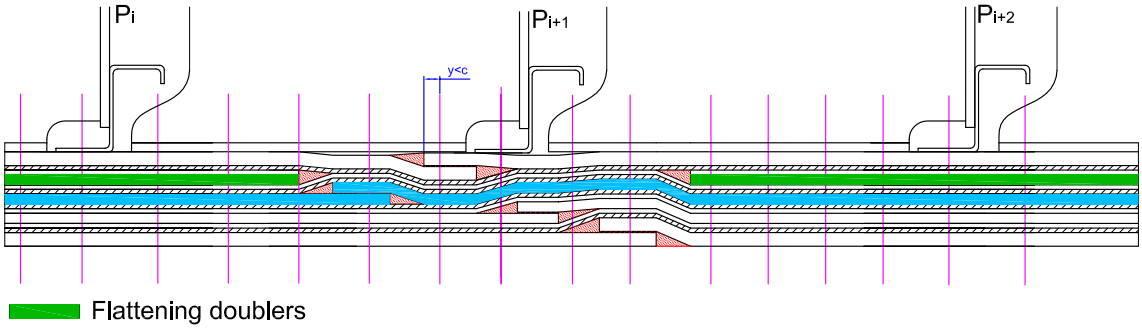


Figure 3.11: Laminate cross-section at C-C.

The splice area and joggles constitute areas in the laminate where no rivets can be inserted, so-called no-riveting areas. The splice area has a minimum width because of strength requirements. The distance between consecutive rivets has a minimum and a maximum value, depending on the their diameter. Finally, the distance between consecutive thickness steps (joggles) has a minimum because of production requirements. It is up to the designers skill to make the back-up structure fit to the laminate, without violating the stated design rules. Solving this problem of often related requirements asks for an iterative approach. Not always can conflicts be prevented and solved in a straightforward manner, which asks for creative solutions. It cannot always be avoided that some concessions to the engineering principles have to be made, as long as airworthiness, producibility and quality are assured.

3.3.2 Conclusions on the design process

The detailed design process for FML fuselage panels is a multidisciplinary process, governed by a large amount of requirements, which have to be implemented during the

detailed product design phase. Because of the high complexity of the product, this implementation often results in conflicts between different requirements. Resolving these conflicts requires an iterative procedure, where detailed information on back-up structure and rivet locations is needed in an early stage of the design process. Finding one solution for the lay-up is difficult enough, leaving little room for further optimisation.

3.4 Expert view on the design problem

Having identified the design problem, and having described the real world it is situated in, the expert domain knowledge of the design problem now has to be formalised. This step is crucial for setting up a well-defined mathematical model of the problem, and for storing the knowledge in the company knowledge repository. First it is discussed where the design problem is situated in the detailed design process. For this purpose, the process of creating the detailed product model will be represented using an activity-relation diagram. In this diagram process steps that can be automated are indicated, as are the process steps that must be done manually. Next a solution approach as developed by the experts to deal with the design problem is formalised. Finally the different entities that constitute the structure and the relations between these entities are described using an entity-relation diagram. Formalising the structural entities and their relations is needed to be able to generate a model of the product. Finally requirements from the different departments on a KBE system, which implements the solution approach, are summarised by means of defining several use-cases.

The Unified Modeling Language (UML) is used to formalise the expert view on the problem. To create a formal model of engineering knowledge, the concepts of entities and process activities, rules and relations, attributes and values, can be used[2]. UML is a modeling language that can be used to graphically represent the expert knowledge based on these concepts, making it possible to capture, store and share the knowledge.

3.4.1 The design process

The activity diagram shown in figure 3.12 indicates two swimlanes, i.e., a manual and an automated, indicating whether a specific activity can be automated or should remain a manual activity. The process of creating a detailed product model is split into two separate process steps, i.e. define practical design and define detailed design, as discussed in section 3.2.2. The process of defining the practical design will have to be done manually, given

the fact that this is the creative part of the process, where the design engineer defines different concepts for the laminate built-up.

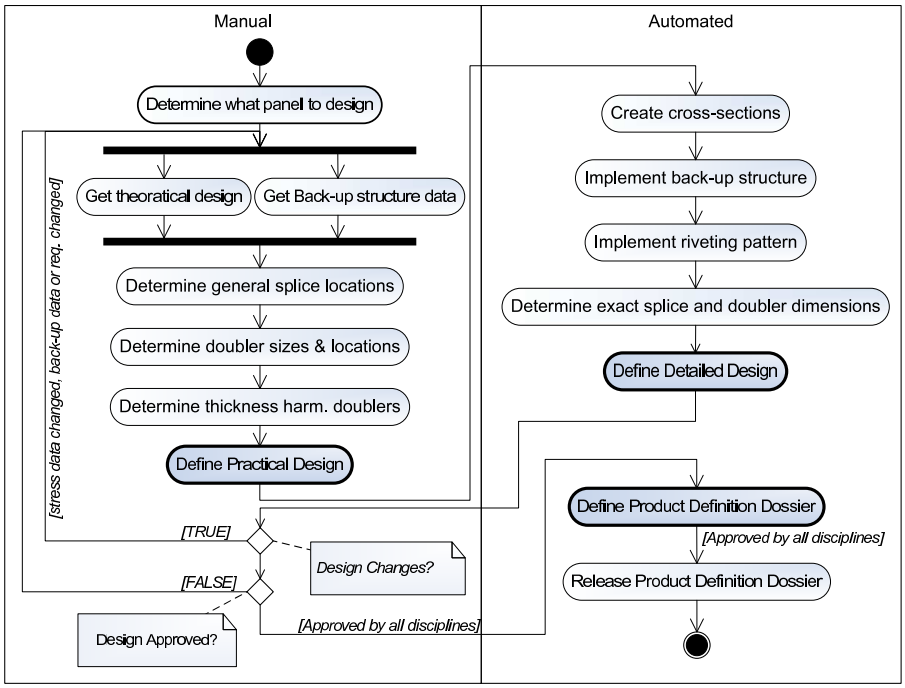


Figure 3.12: Activity-relation diagram of the design process.

The challenge in the detailed design of FML structures, as presented in section 3.3 is in the 'determine exact splice and doubler dimensions' step. Besides the fact that this individual step is iterative, it can be clearly seen that the total process can be iterative. This is caused either by changes in input (stress or back-up data) or requirements (design principles), or by disapproving of the design by the different disciplines involved.

3.4.2 Solution approach used by the experts

In order to develop an efficient solution approach, the expert knowledge on how to find solutions has to be analysed and formalised. For this reason a simple design problem is chosen, consisting of a number of rivets between two consecutive no-riveting areas, see figure 3.13. As discussed, the distance between adjacent rivets has a minimum value d_{min} and a maximum value d_{max}, depending on the the diameter. Figure 3.13 shows the number of rivets that can be instantiated between the two no-riveting areas, located at a distance $L_{pattern}$.

$L_{pattern}$ has to be set to such a value that the riveting requirements are fulfilled. To this

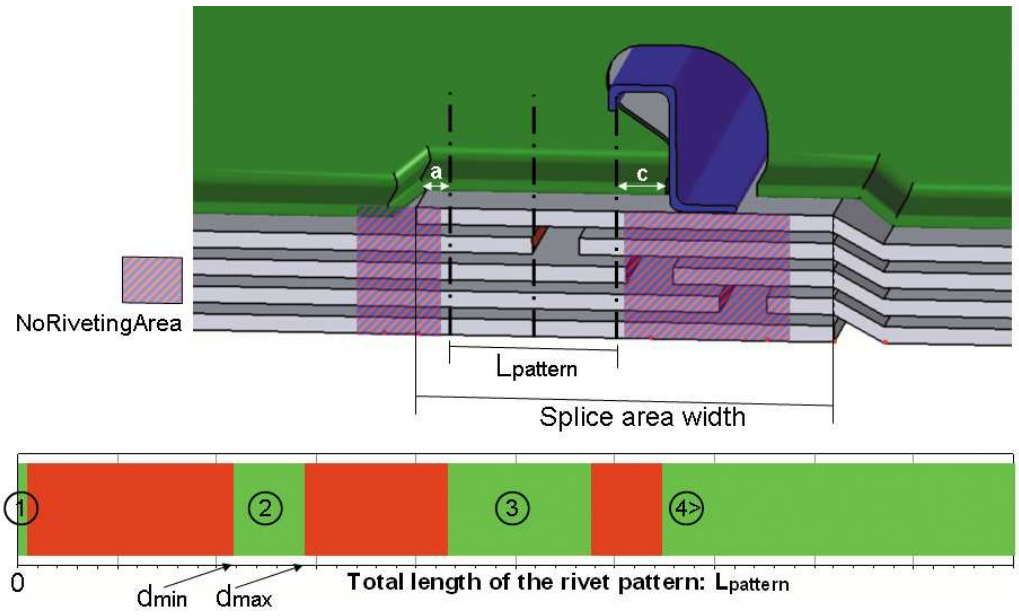


Figure 3.13: Number of rivet instantiations for a given distance $L_{pattern}$ between two no-riveting areas. Red indicates that the rivet requirements are violated, green that they are fulfilled

end the design engineer can increase the width of the splice area or alter the location of the splice area relative to the stringer. Doing so, the no-riveting area related to the splice area will move. The stringer constitutes a no-riveting area for the frame rivet pattern, and since the location of the stringer is fixed, this no-riveting area is fixed.

As shown in the case study, rivet patterns at different cross-sections need to be analysed at the same time, to get a solution for the splice dimensions. Figure 3.14 shows an N^2 diagram of the different no-riveting zones (NR) and riveting patterns (RP) in the case study example of section 3.3. The feedforward relation between for instance $NR1$ and the $RP1$ is a feature based relation, where changing the location of $NR1$ will directly change attributes of $RP1$. A feedback signal is given by $RP1$, containing information on whether the riveting requirements are fulfilled, and what change is needed in the attributes of $NR1$ to accomplish this. The feedforward / feedback signals thus define a bidirectional functional relation, stating that a change in a $NR1$ attribute induces a change in a $RP1$ attribute, and it should be verified if the $RP1$ can still function according to the requirements. Furthermore a cross-coupled change propagation exists between $NR1$ and $NR3$. The relation between the two NRs is a functional relation, since it should be evaluated if the distance between the NRs is according to engineering principles.

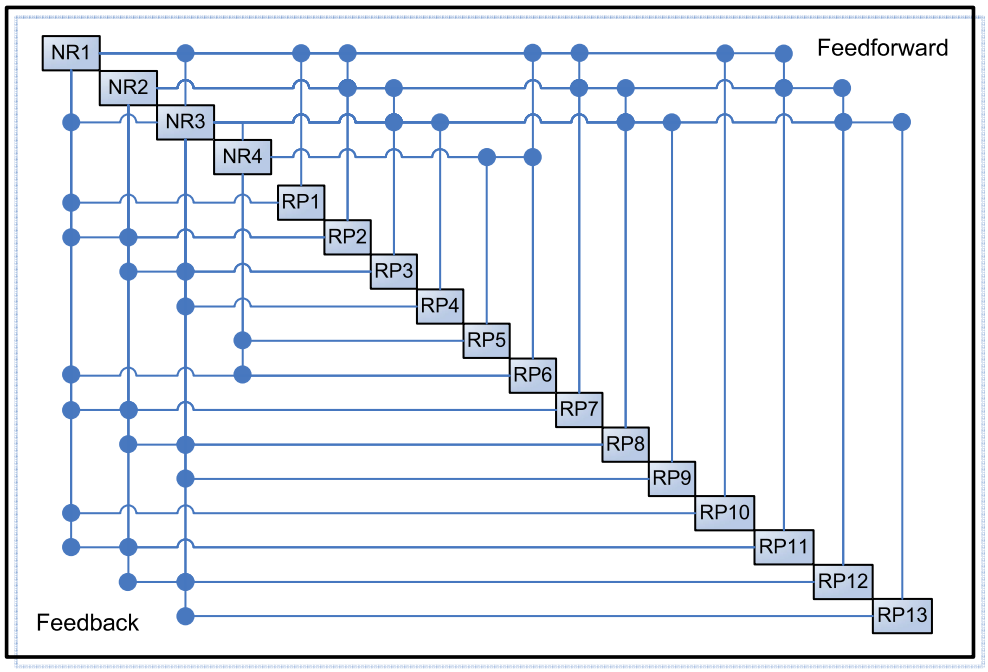


Figure 3.14: N^2 diagram of relations between no-riveting areas (NR) and rivet patterns (RP).

The design engineer decomposes the problem, not to have to analyse all the rivet patterns in the panel at once. The problem can be decomposed by grouping the patterns that have a relation to the same no-riveting area, and solving for this single no-riveting area. Combining the solutions for the decomposed problems is then an iterative process of trial and error.

3.4.3 The product model

The entity diagram in figure 3.15 shows the different structural entities, that should be implemented in the product model, and their relations and requirements. As discussed, a FML fuselage product consists of a skin (the laminate), the backup structure and rivets for joining the backup structure to the skin. The backup structure is composed of frames and stringers, and both can be joggled. The laminate is composed of a basic laminate entity, entities describing the additional layers added to increase the thickness, and finally the splice area entities.

The three entities of which the laminate is composed, can be represented as a general zone entity. This zone entity describes a rectangular section of the laminate with a specific number of layers and a specific type of build-up, e.g. basic laminate or splice area. These

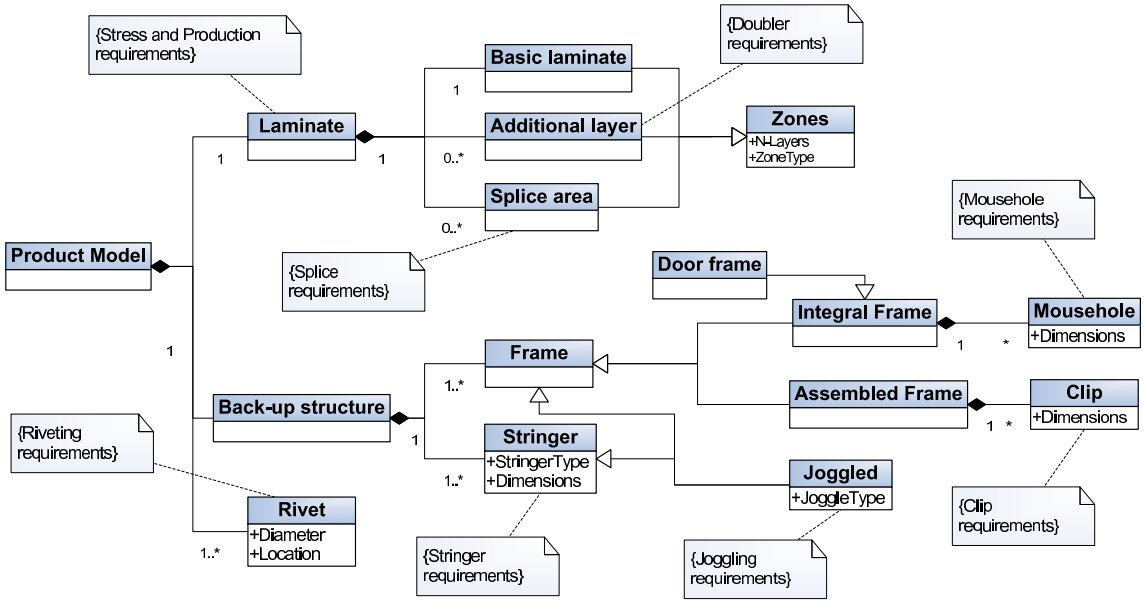


Figure 3.15: Entity-relation diagram of a general FML fuselage product.

rectangular zones together form a grid, describing the properties of the laminate at each specific location, see figure 3.16. The larger the amount of zone entities, the more complex the laminate built-up is.

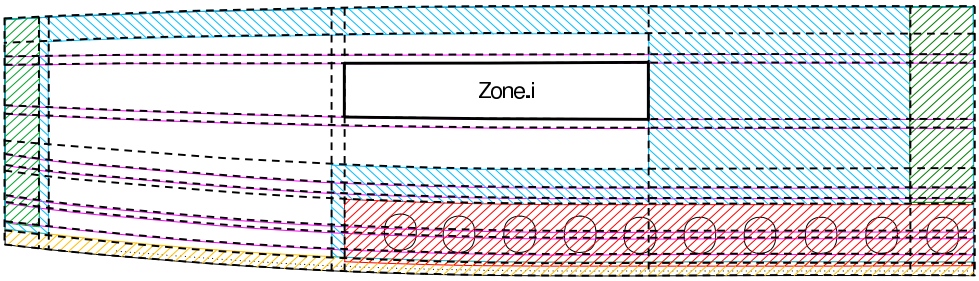


Figure 3.16: Laminate representation using different zone entities.

3.4.4 Requirements on the KBE system

Once a solution approach has been developed, it can be implemented in a KBE system, which will be used by different actors from different disciplines. Each discipline has requirements with respect to functionalities available in the KBE system. These requirements can be mapped using so-called use-cases, describing how a certain actor would make use of the system to achieve his goal, see figure 3.17.

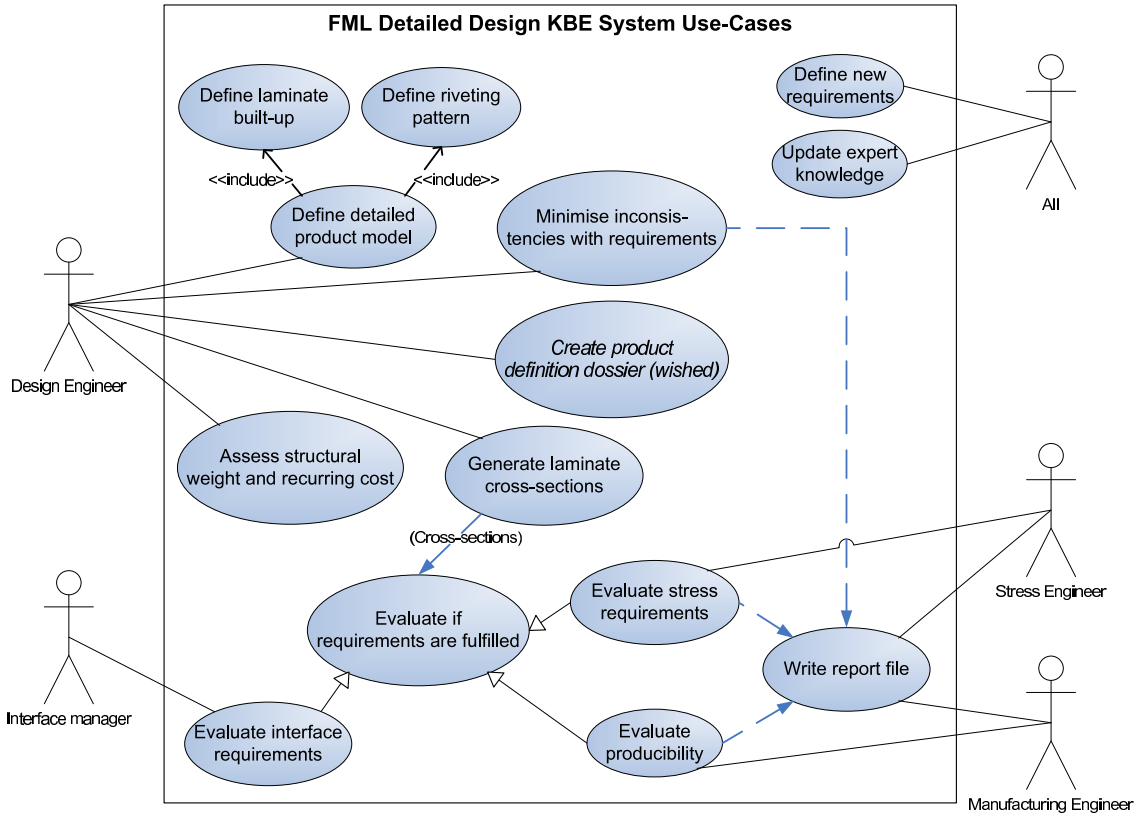


Figure 3.17: Requirements on the KBE system are extracted by defining use-cases.

The design engineer use-cases focus on the task of defining a product complying to all requirements, and generating design outputs. First he needs to generate a product model of the FML panels, and adapt the product variables in order to comply with the requirements. The desired design outputs consist of laminate cross-sections, needed for the other departments to be able to evaluate if their requirements are fulfilled. Finally the design engineer should be able to assess the values of the product attributes structural weight and recurring cost. The other discipline engineers involved in the process, i.e., stress manufacturing and interface management, need to evaluate if their requirements are fulfilled, and write a report containing the results of this evaluation. The KBE system needs to generate information that enables them to do so. The laminate cross-sections for instance are used by the disciplines to assess if the design meets their requirements. Finally, important for the re-usability of the KBE system is the possibility to update the knowledge base and to be able to define new requirements, represented in figure 3.17 by a use-case for all departments.

3.5 Conclusions

The detailed design of FML fuselage structures is dominated by a large amount of detailed requirements from the different disciplines involved. Implementing all requirements in a feasible product definition asks for a large knowledge of the engineering principles, and detailed information on laminate built-up, back-up structure and rivet locations.

An expert view on the design problem is presented, showing the entities of the product and the way they are related. The expert solution approach is to first define areas where no rivets are allowed, so-called no-riveting areas. Next rivets are grouped in rivet patterns and inserted in between two no-riveting areas. By moving a no-riveting area, the designer makes sure that the design principles acting on the rivet patterns will be fulfilled. Finding a solution for the entire panel requires an iterative solution finding procedure.

First step in automating the solution finding is to fill in the N^2 matrix of no-riveting areas and rivet-patterns, and to identify the type of relation between the elements in the matrix. Next a solution finding algorithm is needed to set the locations of the no-riveting areas, complying with all requirements.

Bibliography

[1] Vlot A., et al.: Fibre Metal Laminates, an introduction. *(Dordrecht, Kluwer Academic Publishers, 2001)*

[2] Stokes M. (on behalf of the MOKA consortium): Managing Engineering Knowledge - MOKA: Methodology for Knowledge Based Engineering Applications. *(Professional Engineering Publishing Limited, Bury St Edmunds, UK, 2001)*

Chapter 4

Heuristic Solution Finding Algorithm Based on Expert Domain Knowledge

In order to develop a solution algorithm to the design problem, a well-defined mathematical model needs to be set up. With the mathematical representation of the design problem, a solution algorithm can be developed, specially tailored for solving the specific problem. For the tailoring of the algorithm, knowledge on mathematical optimisation techniques is used in combination with expert knowledge on how to efficiently find solutions to the problem. Efficiently in this chapter is expressed in terms of reduction in computational complexity of finding solutions to a design problem. The computational complexity of an algorithm is defined as a measure of how many evaluations are required in the worst-case in order to find a solution to a given problem [1].

Section 4.1 presents the mathematical model, based on the expert view on the problem presented in chapter 3. Next the mathematical knowledge needed to define a solution algorithm is presented in section 4.2, and the current expert solution approach is presented in a mathematical format. Both knowledge domains are then used to develop a solution algorithm, capable of finding solutions to the problem (section 4.3). Finally the efficiency of the solution algorithm in terms of computational complexity reduction is evaluated in section 4.4.

The following statements will be addressed:

- The FML design problem can be characterised as a Constraint Satisfaction Problem.

- Heuristics can be used to reduce the complexity of the solution finding procedure, without simplifying the problem.

4.1 Mathematical model based on the expert view on the problem

Chapter 3 showed an entity-relation diagram of the design problem. The complexity of finding a solution to the design problem is determined by the number of entities, since they determine the number of design variables and constraints. Table 4.1 shows the order of magnitude of the structural entities involved.

Table 4.1: Order of magnitude of the structural entities

Entity	**Order of magnitude**
Zone	$O(200)$
No-riveting area	$O(30)$
Frame	$O(15)$
Stringer	$O(15)$
Rivet	$O(1000)$

This section discusses what the design variables in the problem are, how the requirements discussed in chapter 3 can be represented as constraints to the variable domains, and what the objective function is.

4.1.1 Design variables

The design variables are the coordinates of all the rivets and the coordinates determining the locations of the quadrilateral zones, see figure 4.1. The location of a rivet is determined by three coordinates in for instance a cartesian system. Since the rivet will be positioned in the panel, a transformation into *panel coordinates* can be made, resulting in two variables for each rivet location R_i (also see figure 4.1):

$$R_i(x, y, z) = R_i(u, v) \tag{4.1}$$

The location and dimensions of a quadrilateral zone is determined by a set of four lines, which have a constant offset with respect to a stringer or frame datum. Thus the zones

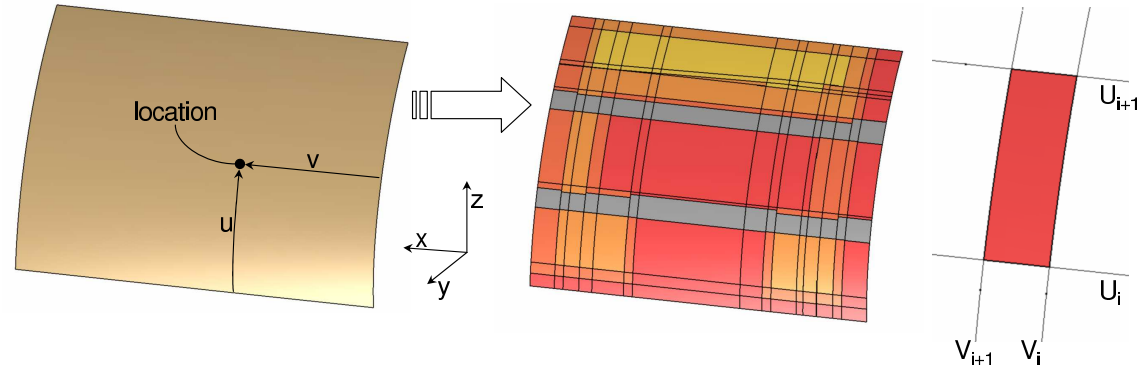


Figure 4.1: Definition of panel coordinates (u,v) and the zones placed in a grid.

are defined by four offset values and accompanying reference datum, see equation 4.2. Z_i describes the location and dimensions of a zone i in the panel:

$$Z_i = Z\left\{\begin{pmatrix}u_i\\P_k\end{pmatrix},\begin{pmatrix}u_{i+1}\\P_l\end{pmatrix},\begin{pmatrix}v_i\\C_m\end{pmatrix},\begin{pmatrix}v_{i+1}\\C_n\end{pmatrix}\right\} \qquad l \geq k, n \geq m \tag{4.2}$$

P and C are given as input by the user, and represent the reference stringer and frame datum respectively. As discussed the zones are forming a grid, describing the laminate properties at each location. The grid has N_{Z_u} zones in the u and N_{Z_v} zones in the v direction.

4.1.2 Constraints

The constraints on the design variables consist of a minimum and maximum distance between two rivets (eq-4.3 **h1**), limitations on the distance between two edges of a zone (eq-4.3 **h2**), and limitations on the position of the rivet within the laminate, with respect to the no-riveting areas (eq-4.3 **h3**). They can be formalised as follows, where $NoRivetArea_j$ describes a surface constituted by a no-riveting area j, and d_{min} and d_{max} define the minimum and maximum rivet pitch respectively:

$$\begin{array}{lll}
\textbf{h1} & d_{min} \leq |R_j - R_i| \leq d_{max}, \quad where\ j \neq i & \\
\textbf{h2.1} & const \leq |u_k - u_i| & \left.\right\} abs(k-i) = 1 \\
\textbf{h2.2} & const \leq |v_k - v_i| & \\
\textbf{h3} & |R_i - NoRivetArea_j| > 0 &
\end{array} \tag{4.3}$$

To be able to select an appropriate solution approach to the design problem, a more

detailed analysis of the constraints is needed. Constraints can be either unidirectional, bidirectional or multidirectional. A unidirectional constraint creates a relation between two variables, where one is determined by the other in a master-slave relation. Having set a value for the master variable, the value of the slave variable will be adjusted in order to comply with the constraint. Bidirectional constraints and multidirectional constraints, involving multiple variables, do not have this master-slave behaviour. These types of constraints can lead to an ambiguous definition of the problem, resulting in several domain sets that satisfy the constraints. Opposite to this possibility is the situation where no solution can be found at all. This can occur when two or more master variables share the same slave variable. Satisfying the constraint for one master variable can result in a violation of the constraint(s) between the slave variable and the other master variable(s), see bold constraints in figure 4.2.

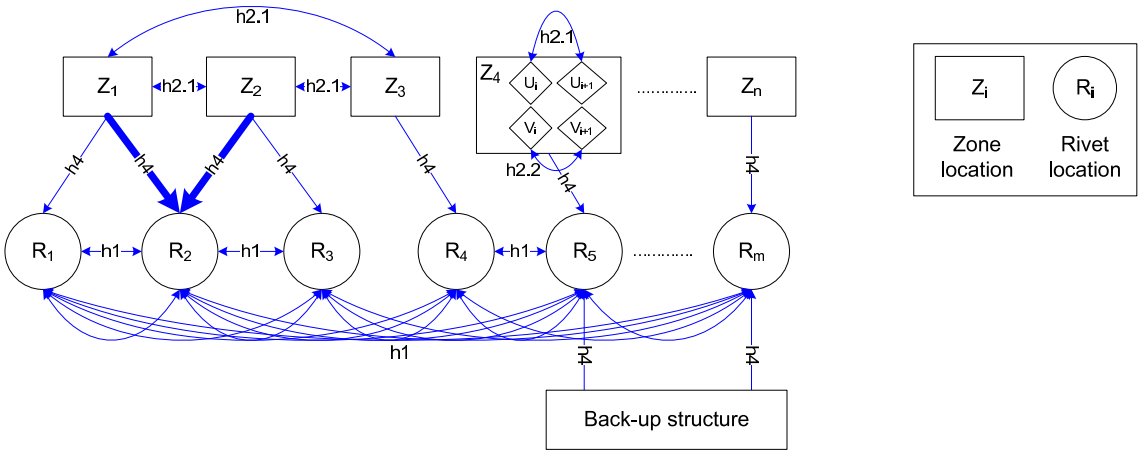


Figure 4.2: Directions of constraints in mathematical model.

Constraint type h1 is a multidirectional constraint on the distance between two rivets. Constraints of type h2 are bidirectional constraints, limiting the distance between the edges of the same or two different zones by a minimum. A zone edge can indicate a thickness step in the laminate, requiring a joggle in the back-up structure. As indicated in chapter 3, the distance between two consecutive joggles is limited due to production requirements. Hence the distance between two zone edges of the same or different zones can be limited by a minimum. Finally, constraint type h3 is a unidirectional constraint, indicating that a rivet should not be positioned in a no-riveting area.

4.1.3 Objective function

As stated in chapter 3, the main challenge in the detailed design of FML products is to find a laminate definition, i.e., assign values to the design variables, fulfilling all requirements. Therefore the design problem can be qualified as a constraint satisfaction problem (CSP), where the goal is to find values to a sequence of variables satisfying all constraints. A CSP, called P, is mathematically denoted as $\mathbf{P} = (\mathbf{X}, \mathbf{D}, \mathbf{C})$, where[2]:

$$\begin{aligned} \mathbf{X} &= \{X_1, X_2, ..., X_n\} && \text{sequence of design variables} \\ \mathbf{D} &= \{D_1, D_2, ..., D_n\} && \text{set of domains for } \mathbf{X}, \text{ where } X_i \in D_i \\ \mathbf{C} &= \{C_1, C_2, ..., C_m\} && \text{set of constraints imposed on } \mathbf{X},\ C_i = g_i(\mathbf{X}) \end{aligned} \tag{4.4}$$

A constraint C_j on a sequence of variables $\{X_1, X_2, ..., X_i\}$ specifies a relation on the allowed combination of values of the variables, resulting in a feasible domain for $\{X_1, X_2, ...$ $..., X_i\}$. A sequence of values $\mathbf{S}= \{S_1, S_2, ..., S_n\}$ is called a solution, if all constraints in $\mathbf{C}$ are satisfied, where $\mathbf{S} \in \prod_{i=1..n} D_i$ [3].

Constraints can be either of an equality type $g(\mathbf{X}) = Const$, or inequality type $g(\mathbf{X}) \leq Const$. A different constraint differentiation is based on the nature of the constraint: in the real-world some constraints can be violated without discarding the sequence of values in $\mathbf{S}$ [2]. A constraint is *hard* if it has to be satisfied. A *fuzzy* constraint allows for a lower degree of satisfaction. *Soft* or *relaxable* constraints can be violated by a solution. Constraints can be *weighted*, meaning that the constraint can be violated, but at a certain cost. Finally, constraints can have a *multi-level* character, which is a combination of the above types. For instance the design principles, discussed in chapter 3, are weighted constraints (guidelines), based on underlying hard constraints (e.g., airworthiness). Implementing this type of constraint involves programming an if-then statement, where the hard constraint becomes activated and a cost is added, *if* the initial weighted constraint is violated, see eq-4.5. This concept of multi-level constraints is common practice in the engineering world, as will be explained in chapter 5.

$$\begin{aligned} C_j &= \{C_{j_w}, C_{j_h}\} && \text{w = weighted constraint, h = hard constraint} \\ cost[C_j(\mathbf{X})] &= 0 && \text{If } C_{j_w} \text{ is satisfied for } \mathbf{X} \\ cost[C_j(\mathbf{X})] &= w_j && \text{If } C_{j_w} \text{ is violated for } \mathbf{X} \wedge C_{j_h} \text{ is satisfied for } \mathbf{X} \\ cost[C_j(\mathbf{X})] &= \infty && \text{If } C_{j_w} \text{ is violated for } \mathbf{X} \wedge C_{j_h} \text{ is violated for } \mathbf{X} \end{aligned} \tag{4.5}$$

A special type of CSP is the over-constrained CSP (OCSP), whose solution *best complies* to the set of constraints. Note that a solution to this type of CSP exists only if not all constraints in **C** are hard constraints. The solutions found depend on the definition of 'best compliance'. Various approaches are discussed in literature [2], all resulting in functions that have to be optimised for 'best compliance'. The approach selected for this specific problem is the weighted approach, where the sum of the cost related to constraint violation has to be minimised:

$$f(\mathbf{C}, \mathbf{S}) = MIN \sum_{j=1}^{n} cost[C_j(\mathbf{S})] \tag{4.6}$$

where f can be seen as a dynamic objective function, having a state that is the summation of the cost of constraint violation. A way of dealing with minimising an objective function in a CSP is to define an additional constraint, stating that the next found solution **S** should result in a lower or equal value of the objective function f:

$$f(\mathbf{S}_{n+1}) \leq f(\mathbf{S}_n) \tag{4.7}$$

4.1.4 Computational complexity of the solution finding algorithm

The order of magnitude of the number of entities in the problem is combined with the above equations to derive the order of magnitude of the variables and constraints in the mathematical representation, see table 4.2. N_{Z_u} and N_{Z_v} are assumed to be order $\sqrt{N_Z}$:

Table 4.2: Order of magnitude of the number of variables and constraints

		Estimation	**Order of magnitude**
	rivet variables	$N_R \cdot 2$	$O(2 \cdot 10^3)$
	zone variables	$(N_{Z_u} + 1) + (N_{Z_v} + 1)$	$O(30)$
h1:	rivet constr.	$\frac{N_R}{2} \cdot (N_R - 1)$	$O(5 \cdot 10^5)$
h2.1:	zone constr.	*typical*	$O(20)$
h2.2:	zone constr.	*typical*	$O(10)$
h3:	rivet location constr.	$N_R \cdot N_{NoRivetArea}$	$O(3.0 \cdot 10^4)$

4.2 Theory on finding solutions to a constraint satisfaction problem

As stated in the mathematical description of the design problem, the number of variables is in the order of magnitude $O(2 \cdot 10^3)$, and the number of constraints is $O(5 \cdot 10^5)$. The problem can be typified as a CSP, containing hard and multi-level constraints. This section presents both mathematical and engineering knowledge on how to find solutions to the design problem. The mathematical knowledge consists of techniques on how to solve a CSP, and how to make the solution finding procedure efficient. By formalising and implementing engineering knowledge on how to solve the design problem, the efficiency of the solution finding algorithm can be improved.

4.2.1 Mathematical knowledge

The solution approach for CSPs can be divided into two general phases, i.e. the propagation and exploration phases. The propagation phase is characterised by removing values from the variable domain $\mathbf{D}$ that do not a priori contribute to a solution to $\mathbf{C}$. This phase is called constraint propagation. In this way the CSP becomes locally consistent, indicating that the remaining values in the propagated domain $\mathbf{D}_p$ are such that they constitute a solution for the variables in each constraint individually [3]. It should be emphasized that the solution is not a global solution. The second phase is the exploration phase, where one or all possible solutions to the CSP are searched for. The propagated domain $\mathbf{D}_p$ is scanned for possible solutions. A tree-search strategy is defined, combining techniques from the field of Operations Research (OR) and heuristics to limit the search without simplifying the problem.

Heuristics

According to Streim[4], heuristic methods use non randomly selected decision operators, which accomplish that potential solutions are ignored. The following general definition of heuristic solution methods is given by Silver[5]: Heuristic solution methods seem likely to yield a reasonable solution to a problem, using experience or judgement, but they cannot be guaranteed to produce the mathematically optimal solution. Both definitions emphasize that when using a heuristic method, it cannot be proven that an optimal solution will be the result.

A general reason for implementing a heuristic solution approach is to better represent the real-world problem, without having to use restrictive assumptions to get to a solution.

Heuristics are hard to be avoided in the case of an explosion of possible combinations of variables, or if an objective function is difficult or very time consuming to evaluate, or when the solution is highly time dependent [5]. By using experience gained during the solution approach or during the solving of other similar design problems, and by judgment or educated guessing whether a chosen direction will yield good results, an efficient problem solving algorithm can be programmed, tailored for the specific problem at hand.

The general approach for finding solutions to a CSP consists of a constructive tree-search. In order to make the search more efficient, several basic types of heuristics are implemented in the search algorithm, i.e. Solution space reduction, decompositioning and constraint relaxation.

Solution space reduction
Basic principle is to reduce the number of solutions that have to be evaluated, without ignoring optimal solutions. Although a very efficient heuristic, this method requires good knowledge of the real-world problem, in order to assess if eliminated solutions will not contain promising results. Some approaches for solution space reduction are [5]:

- Eliminate solutions that don't comply with constraints, for instance through constraint propagation
- Tightening constraints or increasing number of constraints
- Finding patterns in optimal solutions obtained from several numerical instantiations of the problem. Such a pattern can be for instance a variable that always has the same value or a constraint that should always be satisfied.

Problem decompositioning/partitioning
Decompositioning is the division of a complex problem into several more easy to solve sub-problems. The decompositioning can be based on the hierarchy or chronology of decisions/activities, or on the level of influence of different decision variables [5]. Having defined sub-problems, they can be solved in a concurrent, consecutive or iterative manner. With the concurrent approach, the sub-problems are solved independently, and the separate solutions are combined in a feasible solution for the original problem. The consecutive approach uses the output of the first sub-problem as input for the next. The iterative solution approach is a combination of both previous approaches. A sub-problem is solved, keeping the remainder of the problem fixed. The solution found for the sub-problem is

implemented in the complete problem, and a new sub-problem is selected and solved. After one loop, this procedure is repeated. An example of this approach is the bottleneck algorithm, ordering the sub-problems in terms of their influence on the objective function [6].

Approximation methods

Approximations can be used to simplify the mathematical model, find solutions for this simplified model, and evaluate the true model using these solutions. Constraint relaxation is an example of an approximation method. In general, relaxation produces a solution, which for a minimisation problem gives a lower/upper bound on the objective function value of the optimal solution [5]. Constraints can be relaxed by approximation (linear representation of non-linear constraints), or by complete ignoring the constraint. Furthermore, a popular form of relaxation is Lagrange relaxation, where the constraints are multiplied by Lagrange multipliers, and added to the objective function. By doing so, the constraints are implemented in the mathematical model by adding a penalty to the objective function per unit violation of the constraint. The resulting solution to the relaxed problem has to be worked back to a solution to the original problem, either by restoring the constraints in steps or all at once [7].

4.2.2 Engineering knowledge

The expert solution approach presented in chapter 3 can be summarised as follows:

1. Group the rivets in rivet patterns. The rivet patterns have as attributes their length $L_{pattern}$ and rivet type. Various number of rivets in the pattern can be selected, and fulfilment of the design requirements can be evaluated using the rivet pattern attributes.

2. Decompose the problem by grouping rivet patterns, whose attributes are influenced by the same no-riveting area, and solve this decomposed problem.

The rivet patterns are instantiated between two no-riveting areas, automatically fulfilling the constraints of type h3. The start of the rivet pattern is linked to a no-riveting area, the end of the pattern is linked to a following no-riveting area. The distance between these two no-riveting areas determines the length of the rivet patterns, see figure 4.3. Since the start and end of a rivet pattern are linked to a no-riveting area, changing a zone variable and thus moving the accompanying no-riveting area, will change the length of the rivet

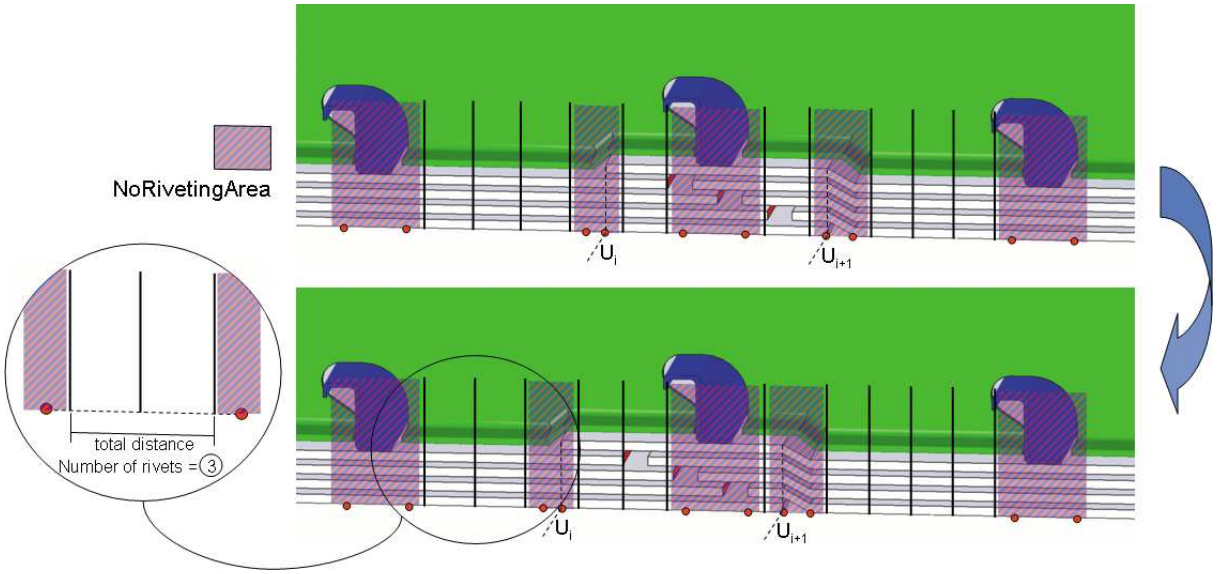


Figure 4.3: Change in rivet patterns, because of changing zone variables and accompanying no-riveting areas.

pattern. The value of the zone variable has to be varied until the design requirements are fulfilled. The aggregation of mutual rivet variables into rivet patterns and aggregation of rivet patterns to zone variables, eliminates all rivet variables. Furthermore, the number of h1 constraints is dramatically reduced by the fact that the total length of the rivet pattern $L_{pattern}$ determines whether the distance between the individual rivets satisfies this constraint. The total number of variables for the design problem is reduced to $O(30)$, see Table 4.1. The number of h1 constraints is reduced from $O(5 \cdot 10^5)$ to the number of rivet patterns, which is typically $O(850)$! A new system of variable / constraint relations is obtained, see figure 4.4.

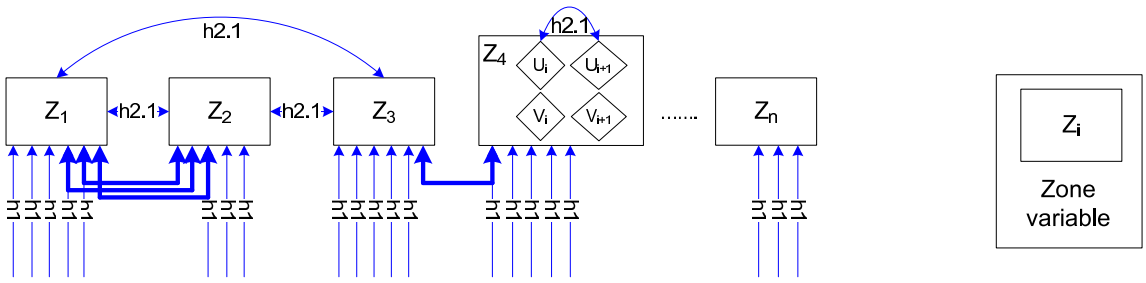


Figure 4.4: Reduced system of variables and constraints, when applying expert solution finding knowledge.

4.3 Knowledge based solution system for the FML detailed design problem

The above described mathematical model and accompanying solution finding approach will be implemented in a knowledge based solution system, whose process flow is described in general in section 2.3. The solution system is based on a modular environment, facilitating a multi-disciplinary approach to deal with complex engineering problems. This approach has been captured in a so-called Design and Engineering Engine (DEE) [8]. The following sections will discuss the solution finding process of a DEE, the generation of the product model and the solution finding phases as programmed in the search engine.

4.3.1 Solution finding process using a Design Engineering Engine

The concepts in the DEE are an input generator, which can be either a designer or a software application or both, a product model generator, different analysis tools, an evaluator and a search engine, see figure 4.5.

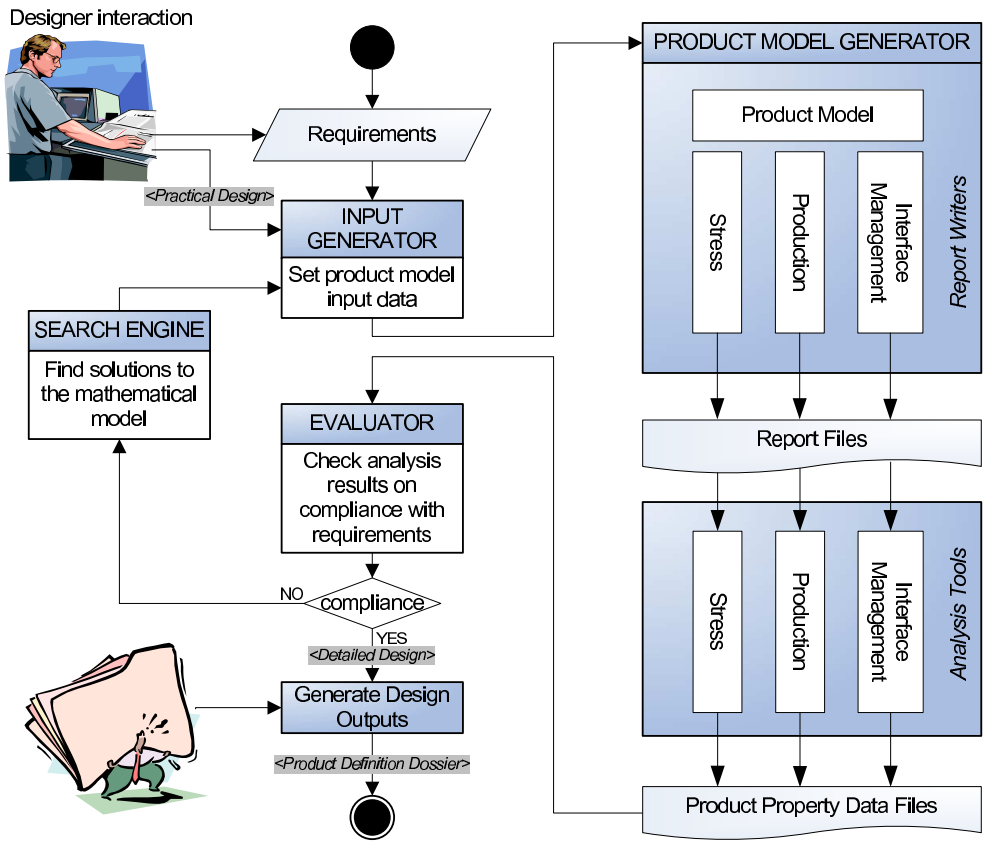


Figure 4.5: FML detailed design process using the concepts from the DEE environment.

The input generator assigns values to the design variables, and stores this data in an input file. This input consists of a zone-wise description of the practical design, manually generated by the design engineer, and detailed backup structure information, see section 3.2.2. Combining engineering knowledge of the product with the information from the input file in an object-oriented software environment, a discipline specific view on the product model is instantiated. This generative character of the product model means that one or more discipline specific views on the product can be instantiated. Section 4.3.2 shows the different steps in instantiating the product model. The product model generates different report files containing product and process information, needed by the discipline specific product analysis tools. The analysis tools analyse the state of the product, describing the behaviour of the product. The evaluator evaluates the state of the product in terms of objective function (f in section 4.1) and constraint satisfaction. If the state is not in compliance with the requirements, the search engine will start looking for solutions to the problem, based on the information exported by the generative product model. Finally the input generator will use the best solution exported by the search engine to change the input file, and the process will start again. The input generator in this design process model is the system user, but he can also decide to allow a coded initiator to automatically alter the variable values according to the best solution generated by the search engine. A more detailed description of the above DEE concepts as implemented in the KBE application for the detailed design of FML panels is given in appendix A.

4.3.2 The product model

The generative product model is build in CATIA V5, a product modeling environment commonly used in aerospace, using visual basic coding and the knowledgeware workbenches available in CATIA V5. Knowledge on how to instantiate a specific panel is stored in the generative model. Given an initial input file of a specific panel, the steps below are followed to create an instantiation of the panel:

1. Instantiate zone primitives, describing the laminate built-up
2. Implement back-up information, such as type of frames
3. Define no-riveting areas in the laminate
4. Instantiate rivet patterns

Appendix A contains a more detailed description of the instantiation of a model of a panel in CATIA. Important in the instantiation process is to notice that the no-riveting areas in the laminate are defined. Furthermore, instead of instantiating individual rivets, they are grouped in so-called rivet patterns, see figure 4.3. Knowing the no-riveting areas, the rivet patterns will not be instantiated in locations where they are not allowed. Therefore the constraint h3 is automatically fulfilled, as stated in section 4.2.2.

4.3.3 Propagation phase

First step in the solution finding process is to apply constraint propagation. Instantiating the rivets in a pattern also influences the riveting constraints h1, since they will no longer apply to individual rivets but to rivet patterns. The new constraints h1 are determined by the total length of the rivet patterns in the panel. Given the fact that the distance between individual rivets is limited by a minimum d_{min} and a maximum d_{max}, ranges of the pattern length where these constraints are fulfilled can be defined, as shown figure 3.13. Constraint h1 can be reformulated according to eq-4.8.

$$\begin{aligned} \textbf{h1} = \text{TRUE if:} \quad & L_{pattern} = 0 \\ d_{min} \leq\ & L_{pattern} \leq d_{max} \\ 2d_{min} \leq\ & L_{pattern} \leq 2d_{max} \\ 3d_{min} \leq\ & L_{pattern} \end{aligned} \tag{4.8}$$

Knowing d_{min} and d_{max} of the specific rivet pattern, the ranges for complying with constraint h1 can be calculated. Values for variable X_i for which h1 is not satisfied, are subtracted from the domain D_i. This is done for all rivet patterns and all variables. By applying constraint propagation, the domain D_i for each variable will be reduced, reducing the amount of evaluations during the exploration phase.

4.3.4 Exploration phase

Having applied constraint propagation, the reduced variable domains can be further reduced by selecting a limited number of values within the propagated domain. Instead of evaluating the four continuous domains for a single rivet pattern as shown in figure 3.13, the domain is discretised using 20 values. This results in a large reduction in the number of values that have to be evaluated in the tree-search. Nevertheless, on average 25 rivet patterns are linked to each zone variable. Although the domain D_i for each variable is

reduced, its dimension is still in the order of $20 \cdot 25 = O(500)$. The number of evaluations of the possible solution vectors is $|D_i|^{N_Z} = 500^{30} = O(10^{81})$, which still results in an impossible evaluation.

To further reduce the number of evaluations, a second heuristic method is implemented, i.e. decompositioning/partitioning of the problem. Figure 4.4 shows constraints h1 applying to different zone variables, creating a bidirectional relation between two variables. Thus in order to correctly evaluate the state of the objective function, these variables should be varied simultaneously. By evaluating the rivet constraints h1, groups of variables that have to be evaluated simultaneously can be defined. These groups are called clusters (also see figure 4.6). The problem is now decomposed, reducing the number of evaluations, $N_{evaluation}$, in the tree search:

$$N_{evaluation} = \sum_{i=1}^{N_C} |D_j|^{N_{Z.i}} = \sum_{i=1}^{15} (500)^2 = O(4 \cdot 10^6) \tag{4.9}$$

where N_C is the number of variable clusters typically $O(15)$, and $N_{Z.i}$ determines the number of zone variables inside cluster i, which is typically $O(2)$. As can be seen the reduction in evaluations is significant, speeding up the algorithm without leaving out feasible solutions.

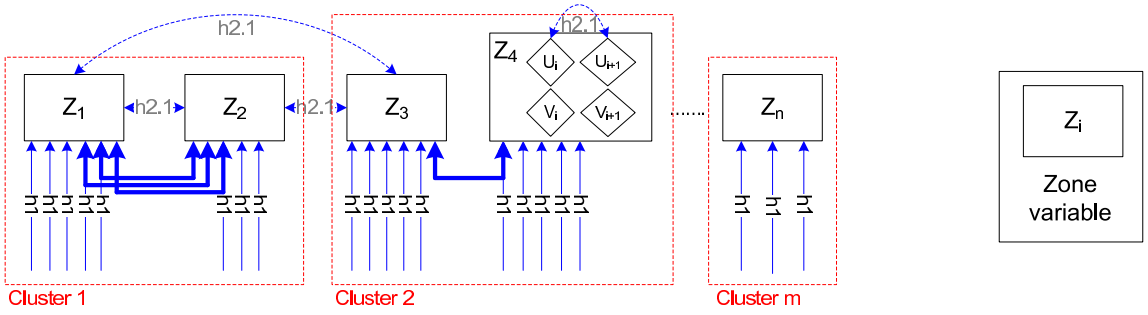


Figure 4.6: Reduced system of variables and constraints as implemented in the solution method.

The above discussed reduction in the number of evaluations due to decompositioning of the problem is only valid in combination with an approximation method called constraint relaxation. As can be seen in figure 4.6, constraints h2 are bidirectional, thus creating relations between the different variables. The relaxation approach taken, is to first ignore constraints h2, making it possible to decompose the problem. When composing the problem, the constraints h2 should be reapplied to the solution vectors of the decomposed problem.

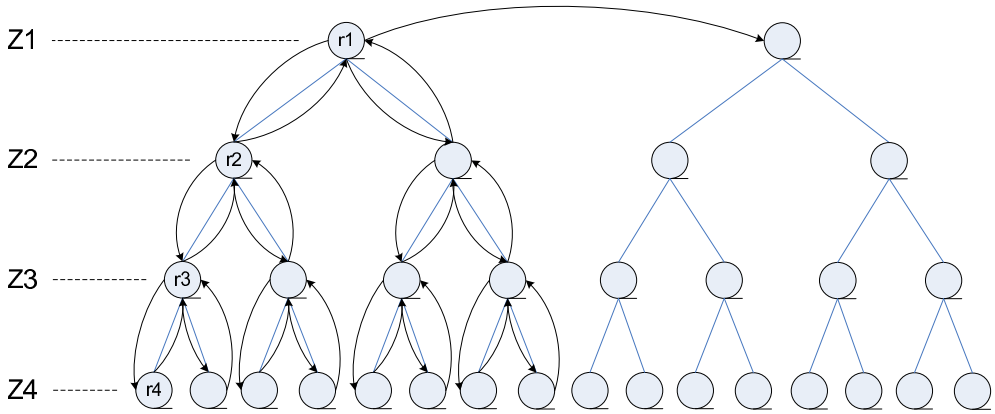


Figure 4.7: Tree search by backtracking the branches of the tree.

The tree search approach shown in figure 4.7 is programmed. Let $\mathbf{Q}$ contain the possible values of the variables, determined by constraint propagation and discretisation of the subdomains D_i. For each cluster of variables, the following backtrack approach is used for finding solutions to the accompanying decomposed problem. Assume a solution sequence to the decomposed problem $\mathbf{R} = \{r1, r2, ..\}$, having a length $|\mathbf{R}|$. The values in $\mathbf{Q}_1$ are candidates for $r1$, and are stored in $\mathbf{Z}_1$ (line5). The smallest element of $\mathbf{Z}_1$ is selected as candidate for the candidate solution sequence $\{r1\}$ (line10) and removed from $\mathbf{Z}_1$ (line(11). The candidate solution sequence is evaluated if it is a solution to the problem (line12) and if so the sequence is stored (line13). Next a sequence of candidate values $\mathbf{Q}_2$ for r2 is stored in $\mathbf{Z}_2$ (line17), and the minimum value from $\mathbf{Q}_2$ is selected as candidate for the candidate solution sequence $\{r1, r2\}$. This procedure is repeated until $\mathbf{Z}_k = \emptyset$. Then we backtrack by refilling $\mathbf{Z}_k$ with the values from $\mathbf{Q}_k$ (line 18) and go one step back (k=k-1, line 19). Next the minimum value of $\mathbf{Z}_{k-1}$ is selected as candidate (line 10), removed from $\mathbf{Z}_{k-1}$ and evaluated if the candidate solution sequence is a solution to the problem, etc.

```
For i=1 To n_cluster
     'Apply constraint propagation on constraints h1 of the decomposed problem
     Q=ConstraintProp(P_i)
     'Perform the tree search
     Z_1 = Q_1
     k=1
     m=1
     While k > 0 Do
          While Z_k ≠ ∅ Do
               {r_k} = MIN(Z_k)
               Z_k = Z_k − {r_k}
               If IsSolution[r_1, r_2, .., r_k] = True Then
                    R_m = [r_1, r_2, .., r_k]
                    m=m+1
               If k<|R| Then
                    k=k+1
          End
          Z_k = Q_k
          k=k-1
     End
Next i

'Compose the problem by applying constraints h2 to R
S=ComposeProblem(P,R)
```

4.4 Evaluation of the efficiency of the solution algorithm

In order to evaluate the efficiency of the solution algorithm, four different test problems are defined and the accompanying computational complexity reduction of the algorithm is determined. The four problems are discussed in appendix B.

Table 4.3: Structural entities of four sample problems

	Rivet patterns [#]	**Zones** [#]	**Frames** [#]	**Stringers** [#]	**NoRivet Areas** [#]
P1	439	54	16	12	20
P2	623	104	16	17	34
P3	492	110	7	18	31
P4	771	214	16	18	33

As stated in the introduction of this chapter, the number of evaluations required to find a solution to a given problem is reflected by the computational complexity of an algorithm. Table 4.3 shows the order of magnitude of the structural entities involved. The resulting number of variables and constraints when instantiating the product model are shown in table 4.4. The complexity of the solution algorithm is calculated in terms of constraint

evaluations. The complexity is calculated as a worst-case scenario for the specific problem, without constraint propagation and decompositioning of the problem. A high complexity for the solution finding procedure is the result, see table 4.4.

Table 4.4: Complexity of the solution algorithm for four sample problems

	$\|\mathbf{X}\|$	$\|\mathbf{C}\|$	$N_{cluster}$ [#]	**Complexity** *Satisfy* **C**
P1	9	129	6	$4.7 \cdot 10^{16}$
P2	14	215	12	$3.8 \cdot 10^{32}$
P3	12	180	9	$5.9 \cdot 10^{24}$
P4	15	245	11	$3.2 \cdot 10^{26}$

The efficiency of the selected solution approach can be measured by the actual number of evaluations when solving the problems compared to the worst-case complexity, see table 4.5 (see appendix B for data generation). A large reduction in complexity of the solution finding algorithm is achieved, mainly due to the fact that a large reduction in variable domain size $|\ \mathbf{D}\ |$ is accomplished by constraint propagation and domain discretisation.

Table 4.5: Worst-case versus actual complexity of the solution finding algorithm for the four sample problems.

	Worst-case Complexity	**Actual Complexity**		
	Satisfy **C**	*constraint propagation*	*optimise decomposed problems*	*compose problem*
P1	$4.7 \cdot 10^{16}$	$1.2 \cdot 10^{5}$	680	$1.7 \cdot 10^{3}$
P2	$3.8 \cdot 10^{32}$	$5.1 \cdot 10^{5}$	138	$5.8 \cdot 10^{5}$
P3	$5.9 \cdot 10^{24}$	$3.6 \cdot 10^{5}$	414	$6.5 \cdot 10^{3}$
P4	$3.2 \cdot 10^{26}$	$3.7 \cdot 10^{5}$	724	$1.3 \cdot 10^{3}$

Bibliography

[1] Hall L.: Computational Complexity. In *(Encyclopedia of Operations Research and Management Science, 2nd edn., Gass S.I., Harris C.M., Kluwer Academic Publisher, pp. 119-122, 2001)*

[2] Meseguer P., et al.(2003).: Current approaches for solving over-constrained problems. *Constraints, Vol. 8, pp. 9-39*

[3] van Hoeve W.J.: Operations research techniques in constraint programming. *(ILLC Dissertation Series DS-2005-02, Amsterdam, 2005)*

[4] von Streim H.: Heuristische Lösungsverfahren; Versuch einer Begriffsklärung. *(Zeitschrift für Operations Research, Vol. 19, pp. 143-162, 1975)*

[5] Silver E.A.: An overview of heuristic solution methods. *(Journal of the Operational Research Societ, Vol. 55, pp. 936-956, 2004)*

[6] Morton T.E., Pentico D.W.: Heuristic Scheduling Systems. *(New York, Wiley Interscience, 1993)*

[7] Patterson R., Rolland E.: The cardinality constrained covering traveling salesman problem. *(Computers and operations research, Vol. 30, pp. 97-116, 2003)*

[8] La Rocca G., van Tooren M.J.L.: Development of Design and Engineering Engines to Support Multidisciplinary Design and Analysis of Aircraft. In *(Delft Science in Design - A congress on Interdisciplinary Design, Faculty of Architecture, ISBN 90-5269-327-7,2005)*

Chapter 5

Activation of the Knowledge Based Solution System in the Engineering World

A knowledge based solution system for the detailed design of FML fuselage structures has been developed, based on the modular approach of a design engineering engine (DEE). The process model of the DEE is discussed in section 4.3. For the user to be able to activate the solution system, a KBE application called ADDET (Automated Detailed DEsign Tool) has been developed. The way the solution system is implemented in ADDET will be discussed in section 5.1. This chapter will furthermore discuss the implementation of the ADDET in the real world, evaluating its effectiveness within the engineering practice. Effectiveness is evaluated by comparing the results with the goals set at the start of the development. First implicit goal of the KBE application is to generate solutions that are useful in the real engineering world, being discussed in section 5.2. Second goal is to reduce lead-time of the engineering process, and reduce the amount of engineers needed, that have the knowledge level required for the design of FML fuselage shells. These goals are being discussed in section 5.3 by means of implementation of ADDET in the engineering process. This section also discusses issues involving the activation of KBE applications in the engineering world.

The following statements will be addressed:

- ADDET generates multiple solutions to the real world design problem.
- A reduction in lead-time and resources can be achieved by implementing ADDET in the design process.

5.1 Implementing the solution system in a KBE application

A KBE application called ADDET has been programmed in an object-oriented language, necessary to capture the data structure (attributes) and behavior (operations) of the knowledge objects discussed in the previous chapters. CATIA V5 has been chosen as graphical modeler, and Visual Basic (VB) as programming language, since an application programming interface (API) is available for VB within CATIA V5. Figure 5.1 shows that the application is composed of a graphical user interface (GUI), the VB code and a primitive library. The GUI layout is based on the use cases of the solution system described in section 3.4.4. The VB code contains the the classes defined in the product model, see section 3.4.3, and contains the different activities described in the design process model, section 4.3.1. More information on the software code and the instantiation of an FML panel in CATIA V5 is given in appendix A.

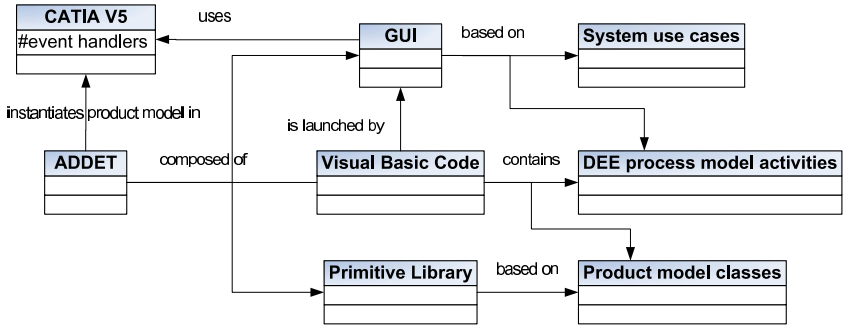


Figure 5.1: Class diagram of the KBE application ADDET.

5.2 Usefulness of solutions generated by ADDET

When evaluating the performance of ADDET, a focus on the usefulness of the generated solutions within the engineering world is needed. This section discusses the solutions generated for the mathematical model of the problem, additional requirements constituted by engineering practice, and whether ADDET can comply with these additional requirements.

5.2.1 Solutions generated for the mathematical problem

A sequence of variable values is a solution for the mathematical problem, if all constraints are satisfied. As presented in chapter 4, some constraints are hard, meaning that the

variable values should be set to satisfy this type of constraint. However, some constraints are weighted, meaning that a violation is allowed but at a certain cost. The objective function in the mathematical model is the summation of the cost of violating the weighted constraints. Table 5.1 shows the number of variables $|\mathbf{X}|$ and constraints $|\mathbf{C}|$, the number of variable clusters $N_{cluster}$, the number of weighted constraint violation $N_{violation}$ and the corresponding value of the objective function.

The four sample problems presented in appendix B are used to evaluate if ADDET indeed generates solutions to the mathematical problem. As discussed in chapter 4, the problem is first decomposed and solutions to the sub-problems are searched. Next the problem is composed, meaning that the solutions to the sub-problem are composed and it is evaluated if they are solutions to the total problem. After composing the problem, different non-related sub-problems can still be identified. The variables in these sub-problems have the same dimension of the solution vector, and are not related to variables in the other sub-problems. Mathematically spoken, the number of solutions of the composed problem would be all possible variable value combinations that are a solution to the problem, i.e., a multiplication of the number of solutions for each sub-problem. From an engineering point of view, however, combining all possible local solutions will not result in significantly different global solutions. Therefore, the number of solutions is given as the maximum of the number of solutions for the sub-problems after composing the problem, see appendix B. Table 5.1 shows the number of solutions $N_{solution}$ generated.

Table 5.1: Results of four sample problems in terms of number of weighted constraints violated, objective function value and number of solutions generated

	$\|\mathbf{X}\|$	$\|\mathbf{C}\|$	$N_{cluster}$	$N_{violation}$	Objective Function	$N_{solution}$
P1	9	129	6	0	0	7
P2	14	215	12	6	1.4	13
P3	12	179	9	10	6.1	24
P4	15	244	11	20	12.7	10

5.2.2 Implementing the multi-level constraint in the engineering world

Chapter 4 presented a mathematical concept of a multi-level constraint. Often hard requirements such as airworthiness or producibility are captured in engineering principles or guidelines. These guidelines help to speed up the engineering process, since the engineers do not have to evaluate all hard requirements, as long as the decisions made comply with

the engineering principles. As stated in chapter 4, an engineering principle can be represented as a weighted constraint, which can be violated but at a certain cost. The cost of constraint violation can be given as a function of the level of violation. This function, from hereon called penalty function, is specified by the engineer and represents a desired value of a design variable based on experience and/or best practice. The concept of a weighted constraint and accompanying penalty function allows the engineer to influence the solution domain, tailoring it for each specific real world design problem.

Four different combinations of hard and weighted constraints can be identified, see figure 5.2. The minimum concept consists of an absolute minimum and a function stating that a larger value of the design metric is desirable. The maximum concept is the opposite of the minimum concept, since it consists of an absolute maximum value and a function indicating that a smaller value of the design metric is wanted. The other two concepts, i.e., subtract and add combination, are combinations of the first two concepts. The subtract combination has a range between the extremes of the weighted functions, whereas the add combination combines the ranges of the weighted functions.

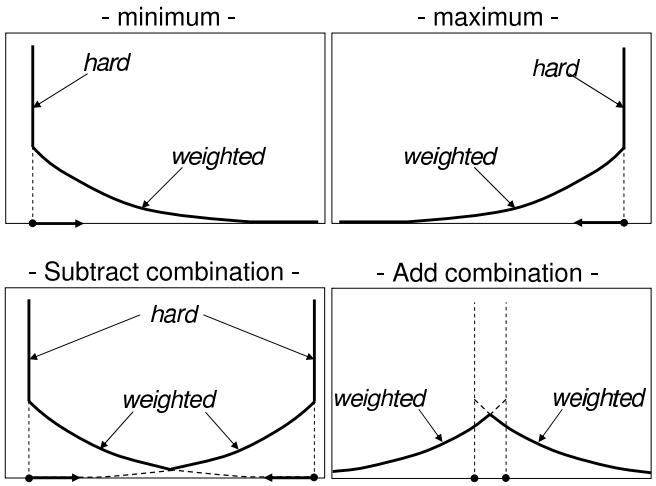


Figure 5.2: Classification of different combinations of hard and weighted constraints.

An example of a design metric that is important for the airworthiness of the FML structure is the rivet pitch. A minimum rivet pitch is defined by the (pin-loaded) blunt notch failure criterion, whereas inter-rivet buckling or the maximum allowable load per rivet constitute an upper limit. The stress department calculates for both criteria and for various load-cases the reserve factor (RF), which is the allowed stress level divided by the applied stress level, see eq. 5.1.

$$RF = \frac{\sigma_{allowed}}{\sigma_{applied}} \geq 1 \tag{5.1}$$

The design principles state a minimum and maximum rivet pitch d_{min} and d_{max}. The value of the penalty function is zero if the rivet pitch is between these two values, see figure 5.3. Decreasing the rivet pitch is allowed, as long as its value doesn't become less than d_{bn}, which is the rivet pitch associated with (pin-loaded) blunt notch (bn) failure. d_{bn} is calculated by setting RF in eq. 5.1 to be equal to 1. The same procedure is applied for the upper limit of the rivet pitch, which is determined by inter-rivet buckling (irb) or the maximum allowable load per rivet. If the rivet pitch is smaller than d_{min} or higher than d_{max}, a new RF should be calculated based on the new applied stress levels. If the resulting RF is larger than 1, the cost of violating the weighted riveting constraint is expressed by the penalty function in figure 5.3. The stress department favours larger over smaller rivet pitches, being expressed in the penalty function by a steeper slope for rivet pitches smaller than d_{min}. If the resulting RF is smaller than 1, then the cost of constraint violation becomes infinite.

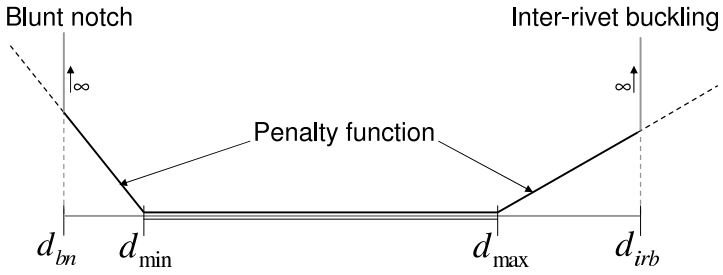


Figure 5.3: Penalty function for the rivet pitch.

Figure 3.12 already showed the number of rivets that can be instantiated according to design principles within a certain distance $L_{pattern}$. Figure 5.4 shows the penalty function for a varying rivet pattern distance. It is a combination of the four penalty function concepts shown in figure 5.2. RF_{old} indicates the reserve factor for a design in compliance with the design principles, i.e., a rivet pitch larger than d_{min} and smaller than d_{max}. The dashed lines show that there is no room for deviation from these limits, since this will result in $RF < 1$. In other words $d_{bn} = d_{min}$ and $d_{irb} = d_{max}$. However, if the old RF is larger than 1, for instance 1.1 as shown in figure 5.4, than there is room for deviation from the limits stated in the design principles. Recalculating d_{bn} and d_{irb} results in a larger feasible domain, at a certain cost however.

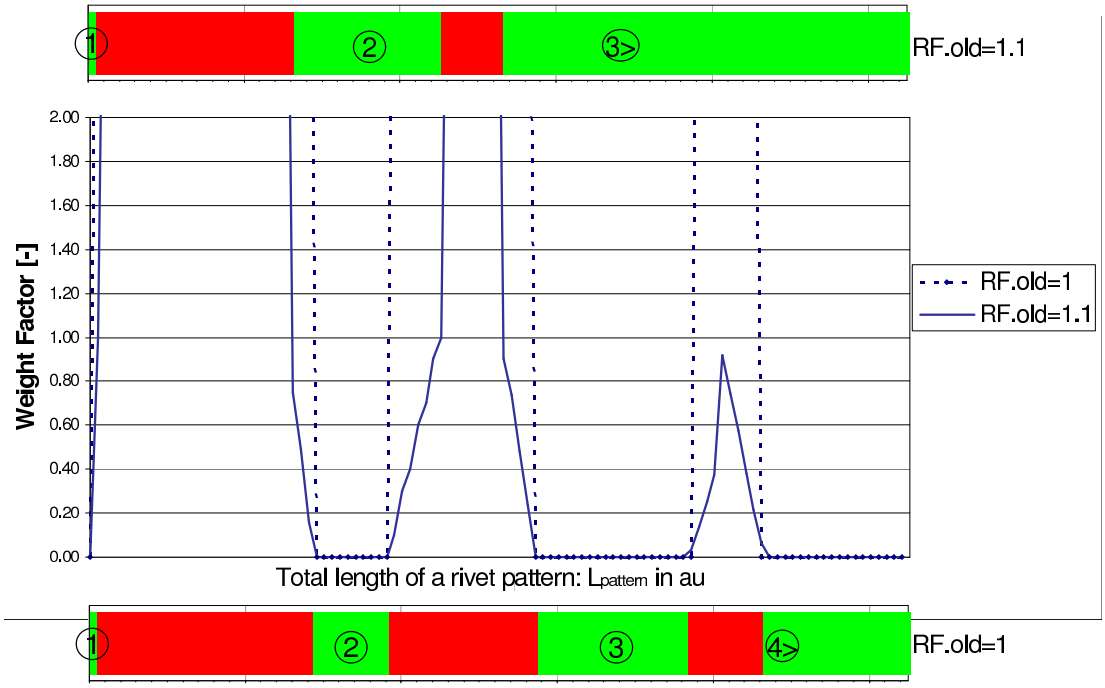


Figure 5.4: Example of a series of penalty functions for constraints on the length of a riveting pattern.

Because of the high importance of the rivet pitch design metric for airworthiness approval, deviations from the design principles should not be accepted without approval from the stress department. In other words, the design engineer can deviate from the design principles when defining the product model, but needs approval from the stress department on each deviation. It should be made explicit by ADDET where a deviation from the design principles is located, and how much this deviation is.

5.2.3 Trade-off between different solutions

Figure 5.5 shows a sample design problem, where a thickness step in the laminate should be positioned in between two stringers. The panel consists of two zones, two stringers and a frame that is connected to the laminate using rivets. The stringer locations are fixed, so the only design variable for this sample problem is the variable X defining the location of the thickness step. The distance between a rivet and a free edge should be equal to c, as discussed in section 3.3.1. The distance p is specified in the design principles. The thickness step in the laminate constitutes a no riveting zone. Furthermore at the location of the stringer-frame intersection no rivets can be inserted.

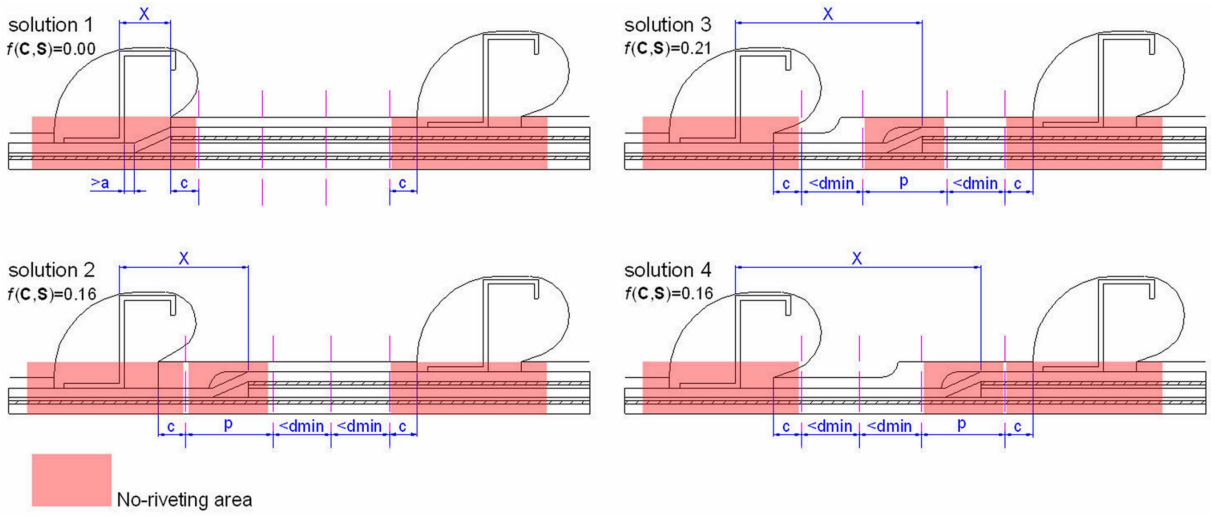


Figure 5.5: Four different solutions to a sample design problem in FML.

The four design options shown in figure 5.5 are all solutions and have been generated using ADDET. Solution 2-4 violate a weighted constraint, stating a minimum distance between two rivets. Nevertheless, they comply with the underlying hard airworthiness requirements, because the for the resulting rivet pitches $RF > 1$. Solution 1 has the lowest weighted sum $f(\mathbf{C}, \mathbf{S})$ and is thus the preferred solution to the sample problem. The other solutions need approval from the stress department, since requirements from the design principles are violated. Using the information on the rivet pitches available in the product model, a report file on the rivet pitches not complying with the weighted constraints can be exported. This overview can be used by the stress department for evaluation and approval.

5.3 Implementing ADDET in the engineering process

Having assessed the effectiveness of ADDET to generate solutions to the engineering problem, the application is activated in the engineering process. The success of the activation depends on the user being able to operate ADDET, and the possibility to update the software in the case of changes in the knowledge base or new required functionalities. Both items require proper documentation of the software. Furthermore, the success of activation is determined by the way ADDET is implemented in the design process flow.

5.3.1 Documentation of the KBE application

A proper documentation of the KBE application consists of the following items:

- Knowledge base and software code documentation
  - Informal model of the knowledge base
  - Formal model of the knowledge base
  - Additional comments in the code
- Activation documentation
  - How to apply in the design process
  - Output validation via a test session
  - User manual
- Software configuration management
  - Hardware and software requirements
  - Software installation and maintenance

Knowledge base and software code documentation
Documenting the code inside ADDET is of critical importance for future use of the application and re-use of the knowledge stored inside the application. During the development of the KBE application, raw expert knowledge has been captured, formalised and finally stored. The concepts defined in MOKA provide a link between raw knowledge, and knowledge that is stored in a KBE application [1]. This transformation of knowledge is facilitated by documenting the knowledge in a special format, suiting the needs of the specific development step. During the capture phase, raw knowledge has to be structured in such a way, that both expert and knowledge engineer can check the knowledge for fitness of purpose. The format is called the informal model, and is designed in such a way that it can be directly linked to the knowledge model used in the next development step, i.e., formalise. The knowledge model used during the formalise phase is called the formal model. The formal model should represent knowledge in such a way that both the application programmer and knowledge engineer can understand the representation, providing a link to the actual computer code. Finally, during the store phase, the knowledge is stored as objects in an object-oriented programming language, which is needed for the computer to understand the knowledge.

Informal model
The concepts in the informal model, as specified in MOKA, are Illustrations, Constraints, Activities, Rules and Entities, in short ICARE forms [1]. Entity forms contain knowledge

on the product, activity forms contain knowledge on the design process, constraint forms contain knowledge on limitations on the product entities, rule forms contain knowledge governing design process activities and illustration forms hold examples, hints, additional general information, etc. By defining relations between these five knowledge concepts, an informal model of the knowledge base is setup. The relations can be represented using different types of charts, e.g., hierarchy and process charts.

Formal model

The formal model consists of a design process model and a product model, based on a refinement of the informal model knowledge concepts and relations. Concepts defined in UML are used to create a formal model, describing a product model consisting of class objects and a design process model consisting of activities. Figure 5.6 shows the formal model consisting of a product and a design process model, and how the informal knowledge units are related to the formal model

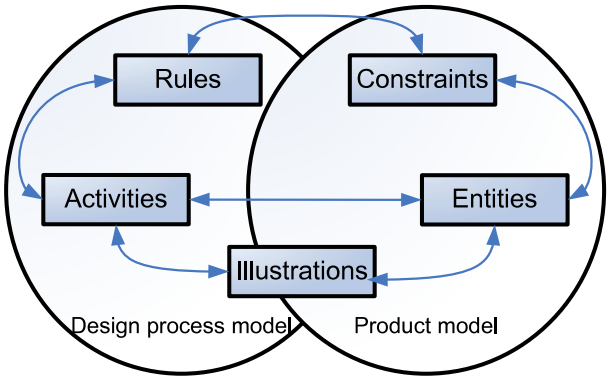


Figure 5.6: Informal knowledge units, used to generate components in the formal model.

Computer code

The classes and activities defined in the formal knowledge model are used to develop the actual code for the KBE application. In order to facilitate comprehensibility and future maintenance of the application, the software code contains detailed comments.

Activation documentation

The design process model contains a description of the new design process, having implemented ADDET. The documentation for activation clearly describes when to apply ADDET in the process, what steps are automated and who should operate ADDET. Section 5.3.2 shows in more detail what the new process will look like, and how ADDET should be implemented in the design process.

In order to prove that ADDET does generate useful solutions to the engineering problem, a test session is defined and documented. The test documentation describes what is being tested, why the test is representative for the engineering problem, and the results obtained.

The KBE application is generic, and is able to provide solutions for many known but also unknown problems. For this reason, the user should be aware of the capabilities and limitations of the application. Besides describing the sequence of steps for a user to take, the user manual should clearly describe the capabilities and limitations of the application.

ADDET provides a platform for the detailed design of FML fuselage panels, as long as the skin can be defined using quadrilateral zones. Panels containing a door cutout can be dealt with, by decomposing the problem in an undisturbed panel and the door surrounding, and then manually composing the problem. This approach is based on best practice, where the laminate in the door surrounding is locally changed to comply with requirements. ADDET is FML specific, since the design rules for FML structures are used; implementing rules for other material families would result in a material independent design platform.

Software configuration management

Configuration management is the discipline of managing and controlling change in the evolution of software systems[2]. An important step in configuration management is to identify the software system elements called configuration items, their version and the relations between the elements. The version identifies the state of a configuration item at a defined point in time. If a version has been tested for consistency and quality, it is made available to the user as a release, and stored in the software repository.

As for the engineering system, the relations define how a change will propagate to the different system elements. Hardware and already installed software are also configuration items. For instance ADDET uses CATIA V5R16 as graphical modeler for the product model; a change in CATIA release can result in a change in API, requiring an update of the VB code (see figure 5.7). Besides a change initiated from within the application, a change can also be requested by a user or knowledge engineer, for instance if a new requirements needs to be implemented, or a new functionality is needed or if the expert knowledge needs updating. These changes will result in an update of the software, executed by the developer. Whether or not all parts of the software needs to be updated depends on the required changes.

If a change propagation results in a change in GUI and/or event handlers, an update in CATSettings is required (see appendix A), resulting in the definition of a new version of

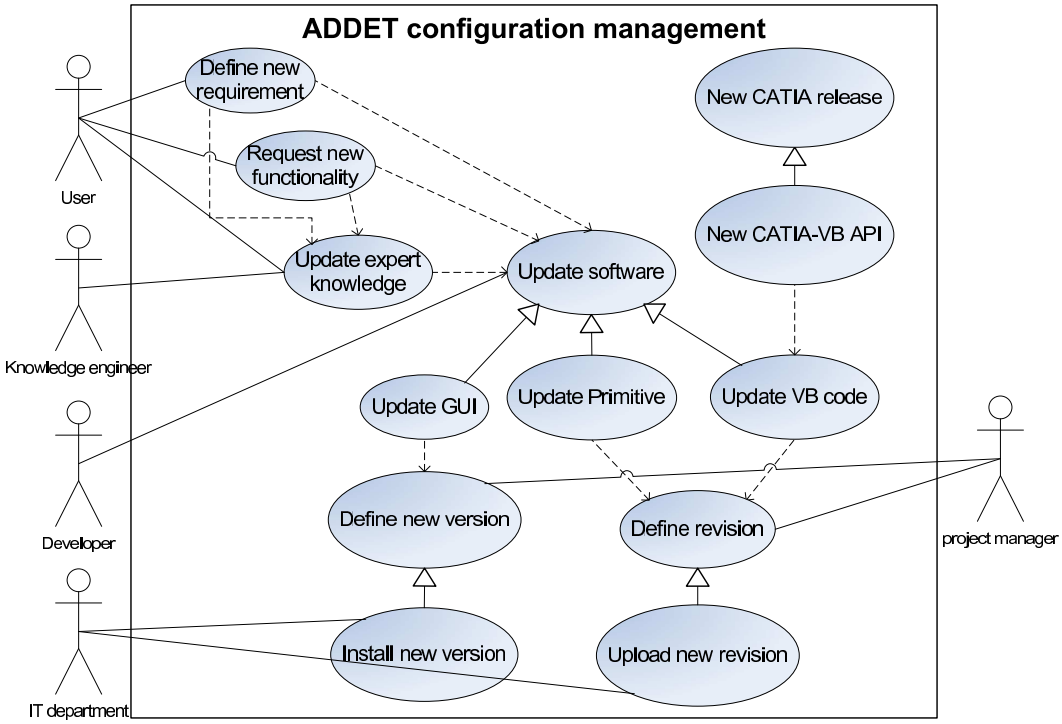


Figure 5.7: Use cases of the configuration management for ADDET.

the software. Changes in VB-script or primitive can be easily implemented by updating the content of the VB-repository or primitive repository respectively, resulting in a new release called a revision. The definition of the new version or revision is the task of the project manager. Finally the IT department installs the new version or uploads the new repository for the revision.

5.3.2 Redefinition of the process flow

As described in chapter 3, the process of detailed product design is executed in a multi-disciplinary environment. In order to identify and implement requirements in an early stage of the process, a concurrent process flow is adopted. The stress department is responsible for the structural sizing of the product. The production department is involved to report production requirements and to check if the design meets these requirements. The interface department is in charge of defining interfaces between products in the assembly and evaluates whether the design complies with these interface requirements. The design department is responsible for the definition of the detailed product model, which should comply with the requirements stated by the individual departments.

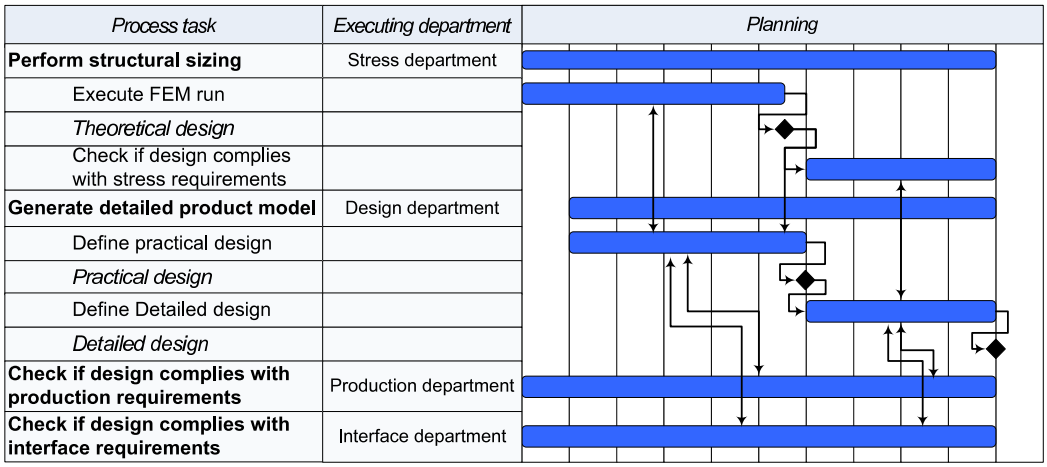


Figure 5.8: Gantt chart of the design process, having implemented ADDET in the design process.

Figure 5.8 shows a Gantt chart of the design process, when implementing ADDET for the generation of the detailed product model of one product. Important difference with figure 3.1 is the division of the generate-detailed-product-model step in two steps, i.e., define practical design and define detailed design. Compared to the traditional process, a reduction in lead-time of 60% for the generation of the detailed product model can be achieved, as is shown in figure 5.9. Besides a reduction in lead-time due to automation of process steps, this reduction is also achieved because of the more sequential approach of the new process. The most time consuming process step, i.e., define detailed design, will start not before the FEM runs have been finalised and a practical design has been defined and approved by all departments involved. This ensures that a complete and stable set of information is used as input for the define-detailed-design step, reducing iterations.

Disadvantage of the Gantt chart representation of a process is the inability to present the number of resources required during the process. For this reason figure 5.10 shows the number of resources on the vertical axis for the different process steps and departments. The number of resources required for the current process of generating a detailed design is shown as a reference.

Two options for planning the amount of resources in the future process can be identified. Option 1 has the same lead-time as the current process. Because of the design time reduction shown in figure 5.9, a reduction in resources required can be achieved. Option 2 has a reduction in lead-time compared to the current process. Figure 5.10 indicates a local increase in resources compared to the number of resources required for the current process. The number of resources deployed depends on company policy for the project. If

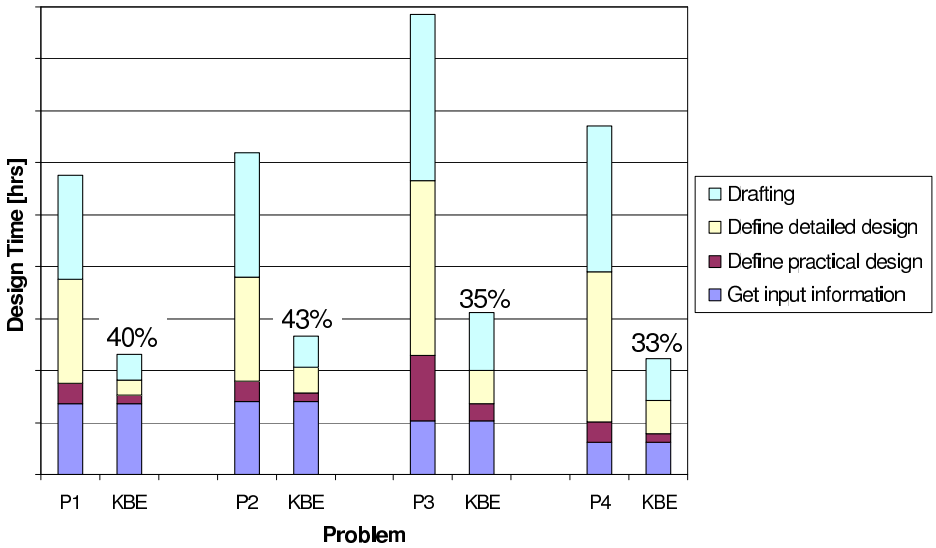


Figure 5.9: Possible reduction of the time to generate the detailed product model using KBE.

the number of design resources is limited, option 1 having a longer lead-time is preferred. If lead-time is a driver for the project, option 2 of deploying more resources for a shorter time frame is preferred. However, it should be noted that this last option also impacts the number of resources required from the other disciplines, since more deliverables that have to be checked will be generated in a shorter time frame. The deliverable release rate is often represented as an S-curve, see figure 5.11. The new release rate will be characterised by a large amount of deliverables being released in a short amount of time, as indicated in figure 5.11.

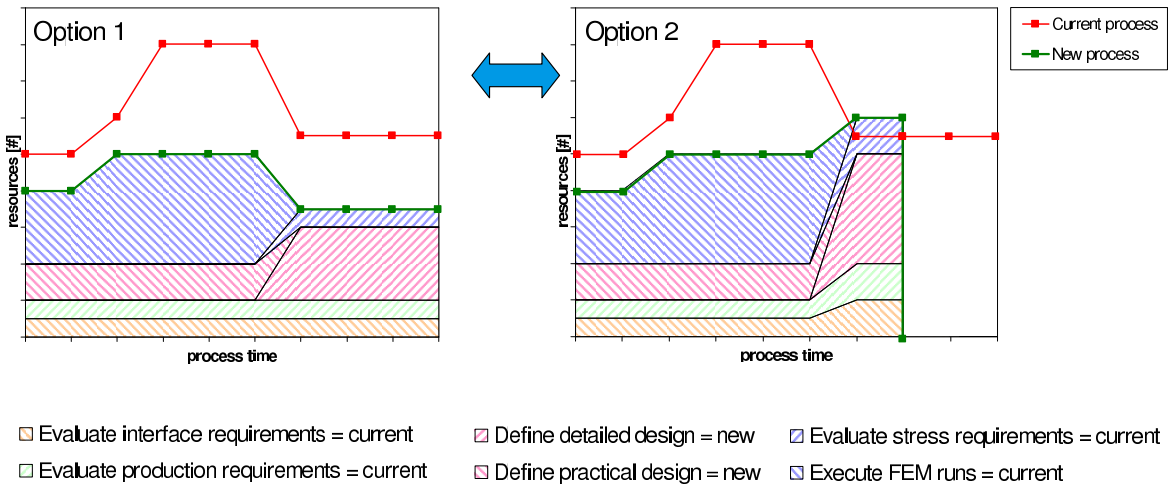


Figure 5.10: Resource planning options when implementing ADDET in the design process.

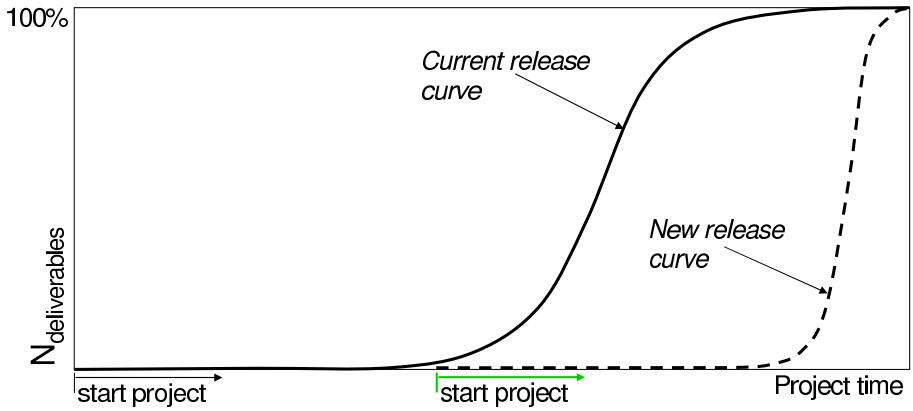


Figure 5.11: Current and future released deliverables curve.

5.4 Conclusions

A knowledge based solution system for dealing with a design problem in the detailed design of FML fuselage skins has been developed and stored in a software tool called ADDET. When implementing ADDET in the real engineering world, it can be concluded that the application is able to generate solutions to the design problem. The weighted constraints, which are based on hard constraints from for instance airworthiness or production, can be violated but at a cost. The cost is defined by the engineer, providing him with the possibility to extend or limit the solution domain. A violation of a weighted constraint is reported to and can thus be accepted by the responsible department on a case to case base.

A reduction in process lead-time of 60% for the steps under consideration can be achieved. When implementing ADDET in the design process this reduction in lead-time can be used to reduce the maximum required amount of resources from the design department. A different option is to reduce the process lead-time, by increasing the amount of resources from the design department. This second option, however, will result in a different deliverable release rate, where a large amount of deliverables will have to be released at the end of the project. This could lead to bottlenecks in the other departments, since they should check a large amount of design outputs in a relative shorter time span. A process wide re-design is needed to prevent these bottlenecks from causing inefficiencies and cancel out the time saved by implementing ADDET. Whether to reduce the maximum amount of resources or reduce the total process lead-time is a company decision, that should be made for each project individually.

Bibliography

[1] Stokes M. (on behalf of the MOKA consortium): Managing Engineering Knowledge - MOKA: Methodology for Knowledge Based Engineering Applications. *(Professional Engineering Publishing Limited, Bury St Edmunds, UK, 2001)*

[2] Bruegge B., Dutoit A.H.: Object-Oriented Software Engineering, using UML, Patterns, and Java™. *(Pearson Prentice Hall, New Jersey, USA, 2004)*

[3] Palmer G.: Java event handling. *(Prentice Hall PTR, Upper Saddle River, USA, 2002)*

Chapter 6

Method for Finding Solutions to Complex Real World Design Problems

A detailed design problem in Fibre Metal Laminate structures has been discussed and a knowledge based solution system has been developed, stored in a software application called ADDET, and implemented in the design process. The development process of the solution system is based on a structured approach, starting with the description of the real world design problem, as discussed in section 1.1. The solution system components are based on the concepts in a design and engineering engine (DEE), presented in chapter 4. The development of the solution system requires a structured approach, combining knowledge from different research areas, in order to tailor the solution system for the complex design problem at hand.

This chapter will present a method for developing a knowledge based solution system, based on a formalised view on the approach used for the development of the solution system described in this thesis. This method will capture and formalise the problem solving knowledge used. A method can be defined as a systematic procedure to achieve an objective. The objective of the method is to find solutions to complex real world design problems. To be able to understand and communicate the steps in the method, a vocabulary is needed. This vocabulary should furthermore help to better understand the objects and relations in the domain, and promote the reusability of the knowledge contained in the method. For these reasons, an ontology of the solution method is presented in this chapter. First the applicability of ontologies for the reuse of problem solving knowledge will be discussed in section 6.1. For the discussion of the ontology, a division of the con-

cepts of the ontology into objects and activities is made. The concepts of the ontology are presented in section 6.2, focussing on formalising the objects in the solution system. The concepts of the activity ontology describe the activities and actors in the development of a solution system, and will be discussed in section 6.3. Finally, some statement with respect to the usefulness of the method ontology will be presented in section 6.4.

The following statements will be addressed:

- Ontologies can be used to formalise knowledge contained in the engineering domain.
- A general problem solving method, applicable to the engineering domain, can be tailored for a specific problem, by mapping its task ontology onto the problem ontology.

6.1 Applicability of ontologies for the reuse of problem solving knowledge

An ontology can be defined as an explicit formal description of concepts in a domain of interest[1]. In general an ontology assigns a domain specific meaning or view to terms, defining a formal domain vocabulary. Ontologies can also be used to formalise knowledge in order to reuse the knowledge, as shown in chapter 3 for instance for the expert domain knowledge. An ontology consists of concepts having attributes and relations between the concepts. The concepts in the ontology are the objects that are present in the domain, and can be for instance physical objects, theories, persons, numbers or process steps. The attributes assigned to the concepts describe the state of the concept. Attributes have a name and a value, and are used to store information on the concept they belong to. Attributes of the concept aircraft for instance can be the range or maximum-take-off-weight. Finally relations define how the concepts interact in the domain. The relation '**is part of**', for instance, indicates an aggregation of a sub-system to a system, like a wing as sub-system **is part of** the aircraft system.

According to the KADS methodology[2], knowledge can be divided in different categories, depending on the type of knowledge being formalised. For each type of knowledge a different ontology typology can be used. First category is domain knowledge giving an expert view on the problem. This type of knowledge is described using a *problem ontology*, and defines the concepts in the domain, their relations, the task set and the solution domain applicable. Second category is the inference knowledge, describing rules of inference based

on the expert knowledge of finding solutions to the problem. Reasoning steps from different expert domains that can be useful for finding solutions to the problem at hand are also contained in this level. This second level of knowledge is represented using an *inference ontology*. The third category contains the knowledge on the sequence of deploying the rules of inference in order to most efficiently find solutions to the problem, contained in a so-called search engine or problem solving method (PSM). This level of knowledge can be represented using a *task ontology*.

To be able to reuse the problem solving knowledge, a problem-independent PSM is needed, which can be achieved by generalisation of different task ontologies. Critics state that due to the presence of the interaction problem, it is not possible to define such a generalisation[3]. This interaction problem states that[4]:

> "Representing knowledge for the purpose of solving some problem is strongly affected by the nature of the problem and the inference strategy to be applied to the problem."

The interaction problem undermines the applicability of a task ontology for developing a PSM for a large array of different problems, since a solution finding method is dependant on problem specific knowledge.

A possible approach for the development of PSM is to start at the problem ontology of the problem at hand. In general a problem ontology consists of concepts having attributes, relations between concepts and an objective or task. According to Reynaud[5] it is possible to define different possible PSMs for the problem by tracing the inference paths, constituted by the relations between the concepts, starting at the objective.

Guarino[6] argues against the interaction problem, by stating that solution finding knowledge is independent of expert knowledge. He states that once a domain has been defined, for instance aircraft engineering, a large part of the solution finding knowledge is similar for various applications. Indeed, the concepts in the aerospace engineering domain, e.g., the disciplines involved and the analysis tools used, are similar for a wide range of problems.

O'Leary[7] states that it could be possible to reuse knowledge through task ontologies for well-structured problems. This well-structured character, however, undermines the gain of reusing knowledge, since a developer can manually setup a solution finding method. Nevertheless, a library of well-structured problem solving methods could save the developer time in setting up the solution system.

Fensel[8] redefines the interaction problem as a tradeoff between usability and reusability. The more reusable a PSM is, the more effort is needed to map the PSM onto the problem domain, in order to make it useful. Fensel states that it is possible to define a domain-independent PSM, which can be reused by mapping its ontology onto the problem specific ontology.

The above discussion clearly indicates the differences in opinion on whether or not it is possible to define a problem-independent PSM, which can be applied to various problems based on its generic character. The knowledge based solution method promoted in this chapter consists of improving the efficiency of inference by combining solution finding knowledge from various disciplines. Selecting the appropriate rules of inference and tailoring them for the problem at hand, is a problem-dependent process. The PSM, defining the sequence of deploying these problem specific rules of inference, can however be formalised in a problem-independent solution finding strategy in the domain of complex engineering problems. In accordance with Fensel's statement, it is possible to define a problem-independent PSM, which can be made problem specific by tailoring it for the problem at hand.

6.2 Task-object ontology of the knowledge based solution method

This section presents the object ontology of the problem-independent method for developing a solution finding system, by discussing the problem, inference and task ontology. As stated in the previous section, it is possible to define a problem-independent PSM, by mapping the task ontology onto the problem and inference ontology. This mapping is translated by means of relations between the three ontologies, shown in figure 6.1. The solution finding strategy (described by the task ontology) is **based on** the solution finding knowledge (described in the inference ontology), and **tailored for** the problem at hand (described by the problem ontology).

First the problem ontology of the domain of complex engineering problems is presented in section 6.2.1. The concepts in the inference ontology are discussed in section 6.2.2. The manner in which the task ontology is related to the problem and inference ontology will be discussed in section 6.2.3. Section 6.2.4, finally, gives a definition of the solution system and the concepts it is composed of.

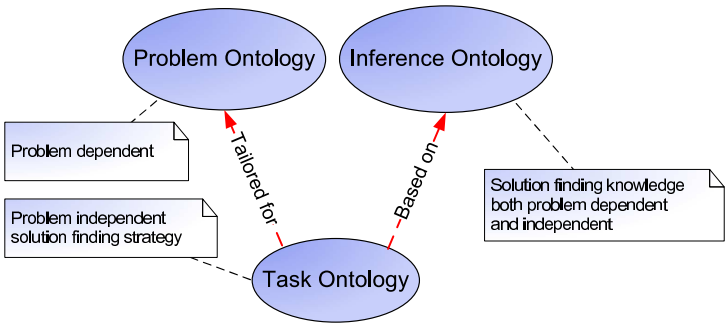


Figure 6.1: Meta-model of the ontology for the solution finding method.

6.2.1 Problem ontology of complex engineering problems

Figure 6.2 shows the concepts of the problem ontology. Expert problem knowledge and expert solution finding knowledge are formalised in a formal expert knowledge base. The attributes of the expert knowledge base are set to be a representation of the real world design problem. Based on information stored in the input file and using the formal expert knowledge base, the product model generator provides the analysis tool with product information. The execution of the analysis is based on knowledge in the formal expert knowledge base. The analysis tool determines the state of the product in terms of objective function value and constraint satisfaction. The analysis tool next sets the attribute values of both the objective function and constraint set accordingly. Besides an objective function and a constraint set, a well-defined mathematical model of the design problem is composed of a variable sequence. The variable sequence has the design variables and their values as attributes, which are stored in the input file. The concepts and relations in the formal expert knowledge base are used to setup a well-defined mathematical model of the problem.

Concepts:

- *Real world design problem.* Informal description of the engineering problem, for which the solution system should find solutions.
- *Formal expert knowledge base.* Formal representation of the expert domain knowledge of the real world problem.
- *Expert problem knowledge.* Formal representation of the concepts and relations in the real world problem.
- *Expert solution finding knowledge.* Formal representation of expert knowledge on how to find solutions to the real world problem.

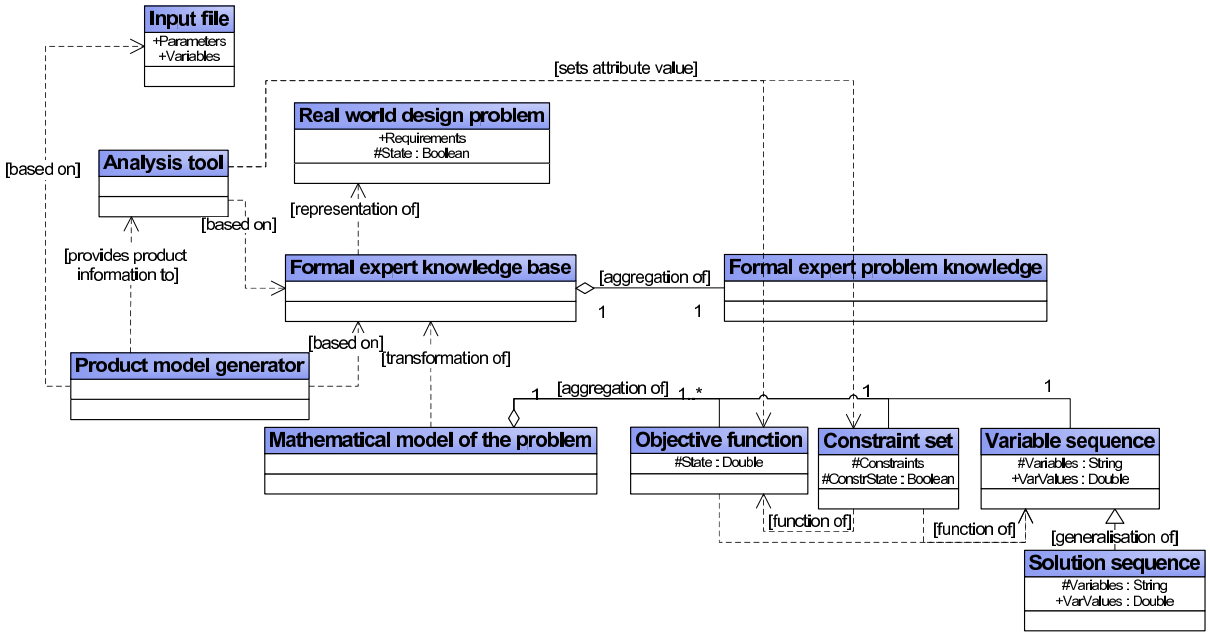


Figure 6.2: Ontology of the problem.

- *Mathematical model of the problem.* Mathematical representation of the expert view on the problem, consisting of an objective function, a constraint set and a variable sequence.
- *Objective function.* State variable of the product model.
- *Constraint set.* Set of equations, limiting the domains of the design variables.
- *Variable sequence.* Sequence of design variables of the design problem.
- *Solution sequence.* Sequence of variable values fulfilling all constraints, optimising the objective function.
- *Product model generator.* Generates a model of the product, based on information specified in the input file and expert product and process knowledge.
- *Analysis tool.* Algorithm calculating the state of the product model.
- *Input file.* Database containing product and process information.

Relations:

- *is representation of.* A manual transformation of knowledge. The relation is executed manually since expert knowledge is not always available in an explicit form.
- *transformation of.* Transformation of knowledge that can be automated, since the concepts and relations in the formal expert knowledge base constitute mathematical equations.

- *based on.* A transformation of knowledge. Without this relation, the concept 'based on' the other cannot exist.
- *provides product information.* An information transfer relation. Automation is possible.
- *sets attribute value.* A data transfer relation. Automation is possible.
- *function of.* A relation between the attribute values of the related concepts defined by a mathematical function.
- *aggregation.* A part-of relation.

6.2.2 Inference ontology of solution finding knowledge

The concepts in the inference ontology are the inference steps contained in expert and mathematical solution finding knowledge. The inference steps in mathematical knowledge are problem-independent and can be for instance an optimisation algorithm, a heuristic or a multiple objective optimisation (MOO) technique. These concepts represent a selection of mathematical solution finding knowledge commonly used in the engineering domain. The inference steps formalised in the expert solution finding knowledge are problem dependent, and are for instance common engineering practice or educated guesses. The relations between both expert and mathematical solution finding knowledge, and the search engine are discussed in the task ontology (section 6.2.3), since these relations constitute the actual mapping of the task ontology into the problem ontology. Figure 6.3 does show the search engine to indicate the relation between expert and mathematical solution finding knowledge.

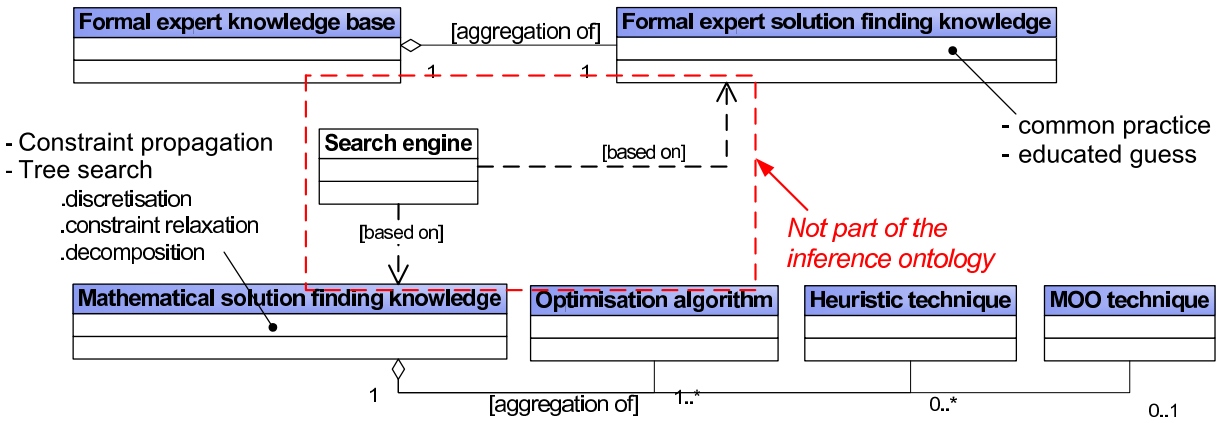


Figure 6.3: Inference steps applicable to the problem domain.

Concepts:

- *Mathematical solution finding knowledge.* problem-independent knowledge on how to solve mathematical problems.
- *Optimisation algorithm.* algorithm to assign values to variables, within a domain limited by constraints , resulting in an optimum value to an objective function.
- *Heuristic technique.* technique on how to reduce the complexity of the problem, without simplification of the problem.
- *Multiple Objective Optimisation (MOO) technique.* technique on how to deal with a simultaneous optimisation of multiple objectives.

Relations:

- *read attribute value.* A data transfer relation. Automation is possible.
- *tailored for.* A knowledge transfer relation. Execution of relation is manual, because of the creativity involved.

6.2.3 Mapping the task ontology onto the problem ontology

The search engine is needed to find solutions to the mathematical model of the problem at hand, and is based on both expert and mathematical solution finding knowledge. In order to make the search engine applicable to the problem at hand, it has to be mapped onto the problem ontology, see figure 6.4. The relation tailored-for expresses this mapping function, where the control of inference step execution is adapted to most efficiently find solutions to the problem. The variable sequence has the design variables and their values as attributes, which are at first set by the information generated by the initiator. The initiator can be either a software application or the system user. When the search engine changes the variable sequence attributes, the initiator reads these new attribute values and changes the input file attributes accordingly. An evaluator has to be defined to check if the state of the product, in terms of objective function value and constraint satisfaction, fulfills the requirements set in the real world design problem.

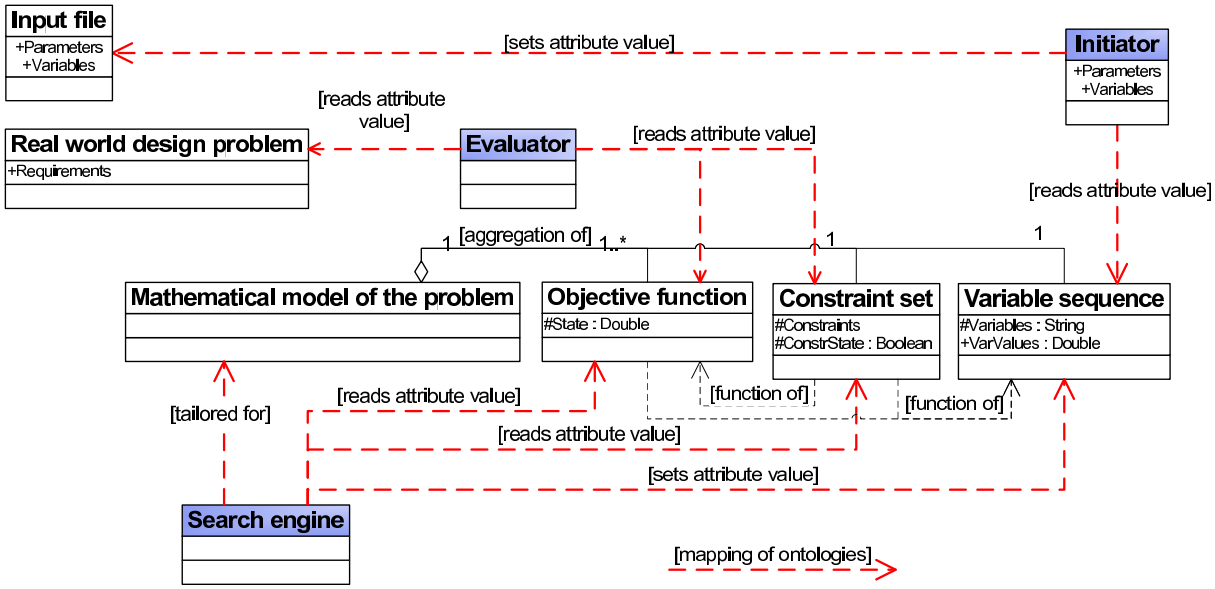


Figure 6.4: Task ontology of the problem solving method.

A problem-independent solution finding strategy is defined, by mapping its task ontology onto the problem ontology. This mapping of ontologies is an activity, hence the relations between the task and problem ontology consist of an activity. This mapping results in a problem specific solution finding strategy, discussed in more detail in section 6.3.

Concepts:

- *Initiator.* Assigns values to the design variables, and reads the attributes of the variable sequence.
- *Search engine.* A concept 'tailored for' the specific mathematical model of the problem to scan the solution domain and find solutions.
- *Evaluator.* Compares the state of the product with the requirements set in the real world design problem.

6.2.4 Solution system ontology

The solution system ontology finally gives a definition of the system, by defining the parts contained in the solution system (figure 6.5). The task of the solution system is to define a solution sequence, whose attribute values constitute a solution to the design problem. The concepts aggregated to the solution system are the same concepts as shown in the DEE in figure 4.5. Finally, the solution system should be activated in the real world, to

assess its effectiveness in solving the real world problem. The solution system will become part of the real world, replacing steps in the real world (design) process using the systems input and output information.

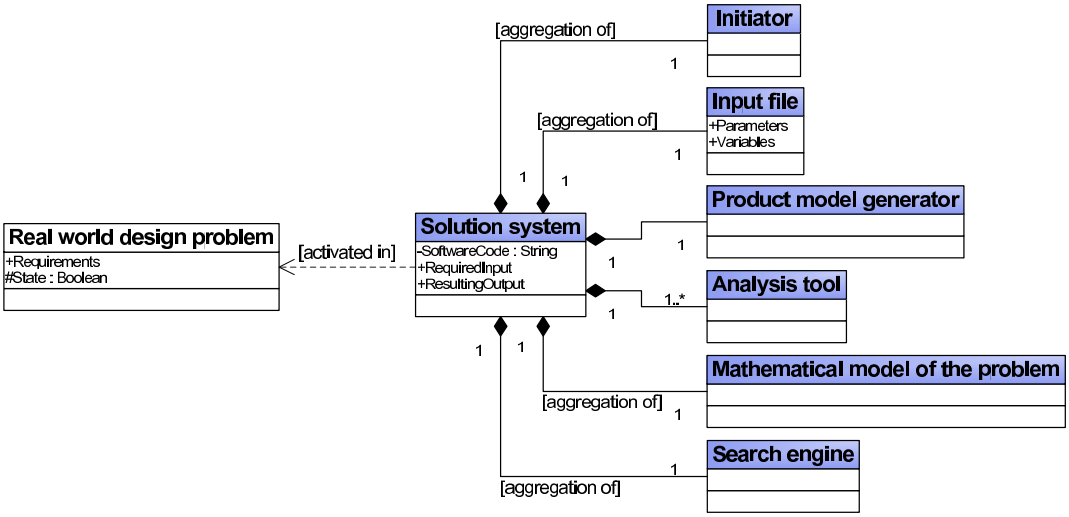


Figure 6.5: Ontology of the solution finding system.

Relations:

- *activated in.* An information transfer relation of input/output information. A concept becomes part of the concept that it is activated-in.

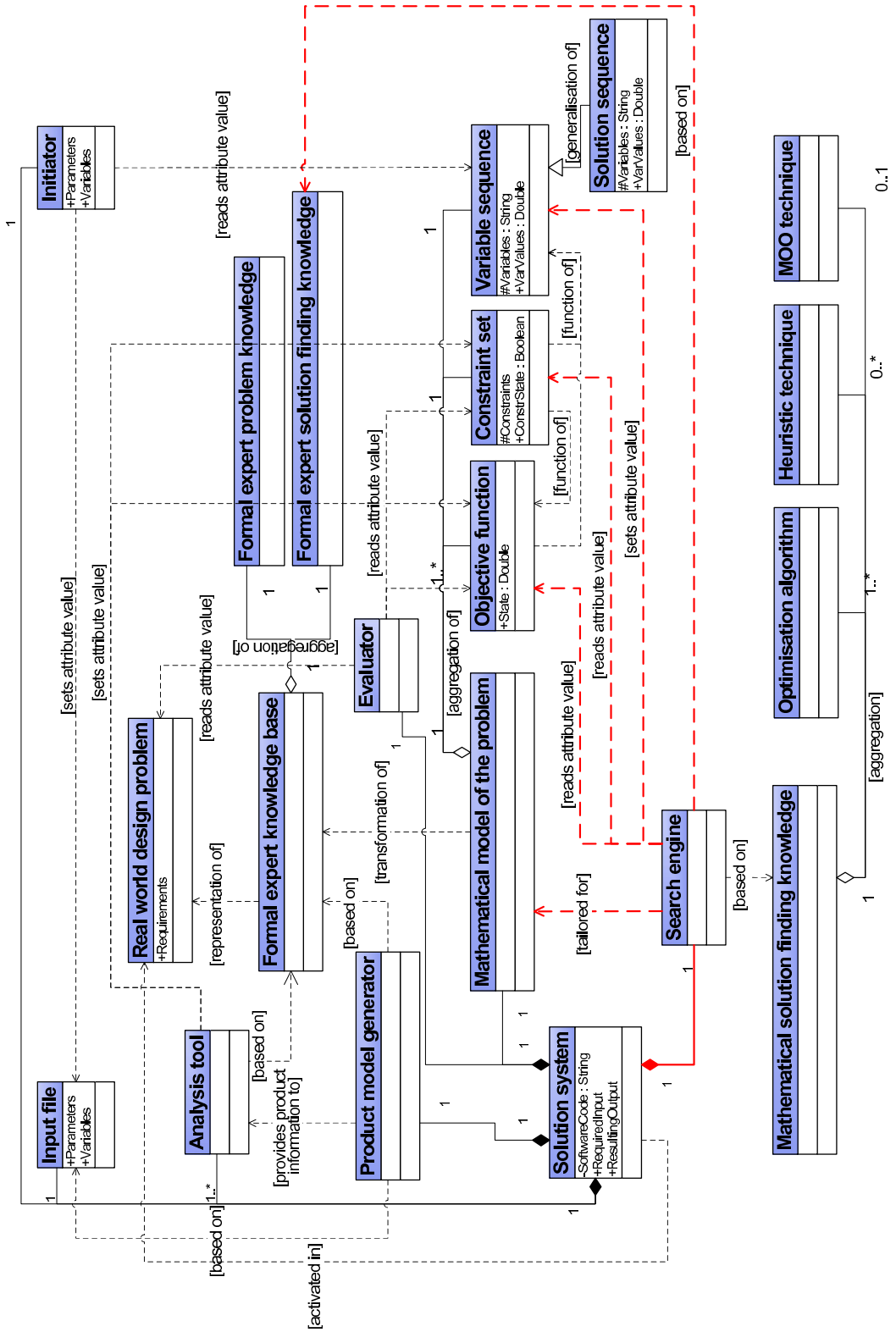


Figure 6.6: Ontology of the solution finding method.

6.3 Activity ontology for the knowledge based solution method

An activity ontology is used to formalise the different steps in a domain-specific process, aimed at simulating the activities[9]. The activity ontology is used to formalise the activities and actors within the method. Figure 6.7 shows the different activity and actor concepts in the process of developing a solution system, tailored to a real world problem.

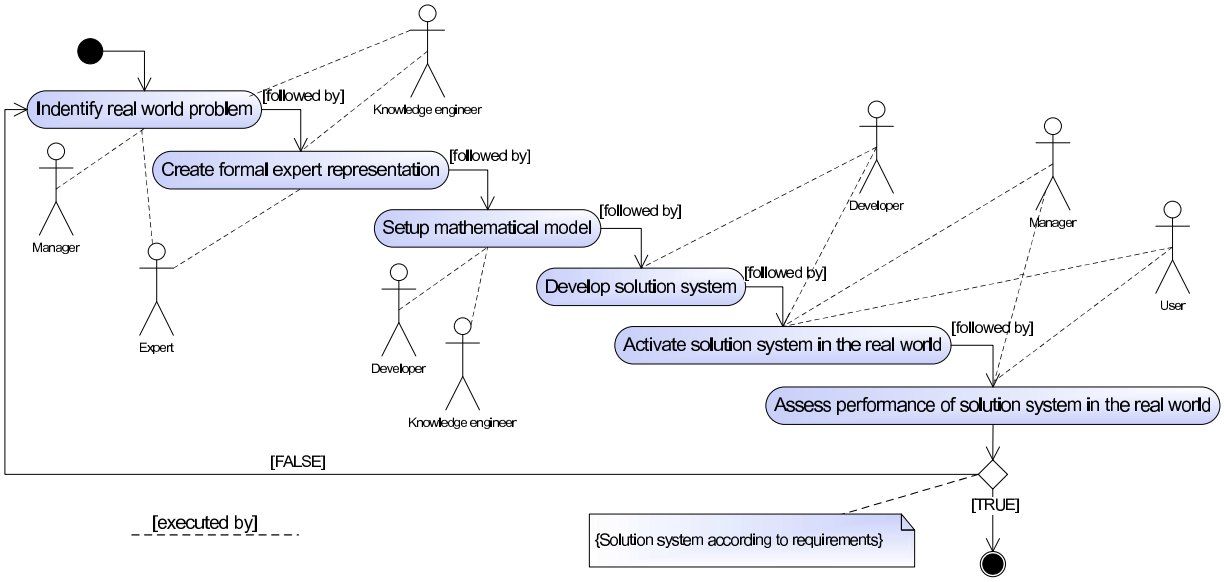


Figure 6.7: Activity ontology of the solution method, describing the different activities and actors involved.

Activity concepts

- *Identify real world problem.* in the real world a problem is identified that needs solving. It has to be assessed if the benefits outweigh the cost of developing the solution system.
- *Create formal expert representation.* the knowledge of the real world problem is formalised by creating and capturing the expert representation of the problem.
- *Setup mathematical model.* This formal expert view is transformed in a well-defined mathematical model, consisting of an objective function, constraints and variables.
- *Develop solution system.* a problem-dependent solution system, based on expert and mathematical solution finding knowledge, is developed.
- *Activate solution system in the real world.* The solution system is activated the real world.
- *Assess performance of solution system in the real world.* The effectiveness of the solution system with respect to dealing with the real world problem is assessed when activated in the real world.

Actor concepts

- *Managers.* Identify the design problem and justify the development of a solution system. They are also in charge of the efficient activation of the system in the real world.
- *Users.* Operate the solution system in order to solve the design problem.
- *Experts.* Have the knowledge on the problem domain, and define which knowledge should be implemented in the solution system.
- *Knowledge engineers.* Investigate the design problem domain, learn what concepts are important in that domain, and define a formal representation of the concepts and relations.
- *Developers.* Transform the formalised domain knowledge in a solution system.

As discussed in the previous section, a problem specific solution finding strategy can be tailored by mapping the task ontology onto the problem ontology. An example of a resulting solution finding strategy is shown in figure 6.8, which is the solution finding process captured in the design and engineering concept, discussed in chapter 4.

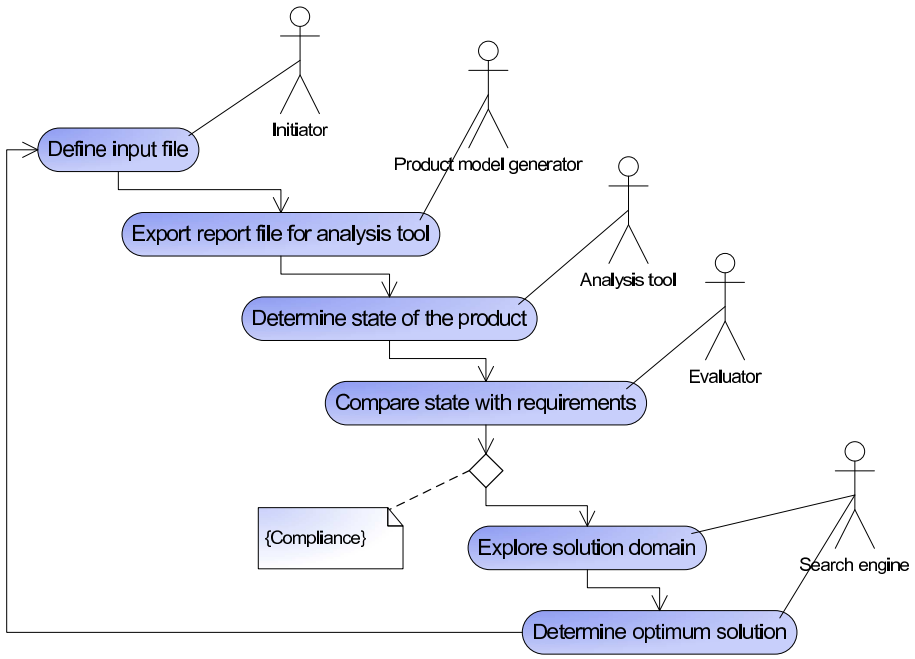


Figure 6.8: Activity diagram of the solution finding process.

6.4 Conclusions

The possibility to define problem-independent PSMs has been the topic of much discussion in literature, achieving no consensus. Fensel[8] summarised the discussion by stating that a tradeoff should be made between reuse of problem solving knowledge and the usefulness of the resulting PSM. He stated that a problem-independent PSM can be defined for specific domains, by mapping its task ontology onto the ontology of the problem. This mapping of ontologies is a problem dependent activity, tailoring the solution finding strategy for the problem at hand. There are numerous problem-independent PSMs used in practice, such as Sequential Quadratic Programming (SQP) or Genetic Algorithms (GA), proving the usefulness of the problem-independent PSM.

The different levels of ontologies, defined by the CommonKADS methodology[2], are very useful for formally describing the concepts in a knowledge based solution system for engineering problems. The mapping of concepts and relations at different ontology levels is the creative part of building the system, and provides its developer with the possibility to tailor the solution system for the problem at hand.

When developing a knowledge based solution system for complex design problem, the following steps should be taken: Identify real world problem, create formal expert representation, setup a mathematical model of the problem, develop solution system tailored for the mathematical model, using available mathematical solution finding knowledge or formal expert problem solving knowledge, and finally assess the usefulness of the solution system by implementing the system in the real engineering process.

Bibliography

[1] Gruber T.R.: Toward principles for the design of ontologies used for knowledge sharing. *(International Journal of Human-Computer Studies, Vol. 43, pp. 907-928, 1995)*

[2] Balder J.R., Akkermans J.M.: Formal methods for knowledge modelling in the CommonKADS methodology: a compilation. *(Petten, Netherlands Energy Research Foundation ECN, ECN-C-92-080, 1992)*

[3] van Heijst G., Schreiber A.Th., Wielinga B.J.: Using explicit ontologies in KBS development. *(International Journal of Human-Computer Studies, Vol. 46, pp. 183-292, 1997)*

[4] Bylander T., Chandrasekaran B.: Generic tasks for knowledge-based reasoning: the "right" level of abstraction for knowledge acquisition. *(International Journal of Man-Machine Studies, Vol. 26, No. 2, pp. 231-243, 1987)*

[5] Reynaud C., Tort F.: Using explicit ontologies to create problem solving methods. *(International Journal of Human-Computer Studies, Vol. 46, pp. 339-364, 1997)*

[6] Guarino N.: Understanding, building and using ontologies. *(International Journal of Human-Computer Studies, Vol. 46, pp. 293-310, 1997)*

[7] O'Leary D.E.: Impediments in the use of explicit ontologies for KBS development. *(International Journal of Human-Computer Studies, Vol. 46, pp. 327-337, 1997)*

[8] Fensel D., Motta E., Decker S., Zdrahal Z.: Using ontologies for defining tasks, problem-solving methods and their mappings. published in Plaza E., Benjamins V. R.: Knowledge Acquisition, Modeling and Management.*(Springer-Verlag, pp. 113-128, 1997)*

[9] Mizoguchi R., Ikeda M.: Towards ontology engineering. *(Osaka University, Technical Report AI-TR-96-1, 1996)*

Chapter 7

Conclusions

The design process of aircraft is becoming more and more complex, due to the increasing amount of design requirements that have to be fulfilled. Implementing all requirements demands for control on the flow of information between different disciplines and between different elements in aircraft (sub)systems. Information transfer between disciplines is often complicated, because different product models are used for analysis, requiring a redefinition of the information. Furthermore, the time consuming process of generating design outputs has to be repeated each design cycle.

The detailed design of fibre metal laminate (FML) fuselage panels is governed by a large amount of requirements from different disciplines. Detailed information on how the back-up structure of the fuselage is joined to the laminate is needed, to such an extend that the location of each rivet needs to be checked for compliance with the requirements. Implementing all requirements in a feasible product definition asks for a large knowledge of the engineering principles, and results in an iterative and time consuming process.

Design steps can be automated once the knowledge needed to execute the step is captured and stored in a consistent format. A generative product model is defined, based on input information and expert domain knowledge. The different disciplines involved receive a set of information needed for their analysis from the product model generator, making the redefinition of information superfluous. The cost of iterations in the process is reduced by automation, making it possible to quickly incorporate input changes and decreasing the lead-time. In addition to lead-time reduction, implementing a knowledge based engineering (KBE) application in an efficient way will also result a reduction in resources by postponing engineering tasks until a complete set of input information is available.

A knowledge based system for solving a design problem in the detailed design of FML fuselage skins has been developed and stored in a software tool called ADDET. The

application generates solutions to real world FML design problems, and reduces the time needed for the detailed design by 60%. This reduction in lead-time can be used to reduce the already limited amount of knowledgable resources needed.

The success of activating ADDET in the engineering world depends on the user being able to operate ADDET, and the possibility to update the software in the case of changes in the knowledge base or new required functionalities. Both items require proper documentation of the software. Proper documentation consists of a formal model of the knowledge base, commented software code, activation documentation and a description of software configuration management.

Automating the generation of design outputs will result in the release of a large amount of deliverables at the end of the project. This could lead to bottlenecks in the other departments, since they should check a large amount of design outputs in a relative shorter time span. A process wide re-design is needed to prevent these bottlenecks from causing inefficiencies and cancel out the time saved by implementing ADDET.

Developing a knowledge based solution system for complex design problems consist of a structured approach of transforming the real world problem, via an expert view on the problem, to a mathematical model of the problem, for which solution finding knowledge is present or can be defined using expert problem solving knowledge from different knowledge domains. Finally the usefulness of the solution system for solving the real world problem should be assessed, by implementing the system in the real engineering process.

A tradeoff should be made between reuse of problem solving knowledge and the usefulness of the resulting problem solving method (PSM). A problem-independent PSM can be defined for specific domains, by mapping its task ontology onto the ontology of the problem. This mapping of ontologies is a problem dependent activity, tailoring the solution finding strategy for the problem at hand.

The different levels of ontologies, defined by the CommonKADS methodology, are very useful for formally describing the concepts in a knowledge based solution system for design problems. The mapping of concepts and relations between each ontology level is the creative part of building the system, and provides its developer with the possibility to tailor the solution system for the problem at hand.

Appendix A

Software Architecture of ADDET

This appendix will present how the concepts in the design and engineering engine (DEE), as discussed in chapter 4, are implemented in a KBE application. The application is called an Automated Detailed DEsign Tool (ADDET). The application consists of a graphical user interface (GUI), visual basic (VB) code and a primitive repository, see figure A.1. The GUI is used by the software user to launch the different VB modules in the code. These modules contain the steps in the DEE process model, discussed in section A.2, and the product model classes presented in figure A.2. For each of the product model classes, a so-called primitive is defined using the user-defined features (UDF) concept in CATIA V5. A UDF is a pre-defined structural entity that can be instantiated in CATIA as often as needed, based on geometrical and dimensional information. The primitives have the attributes and behaviour (operations) of the classes in the product model.

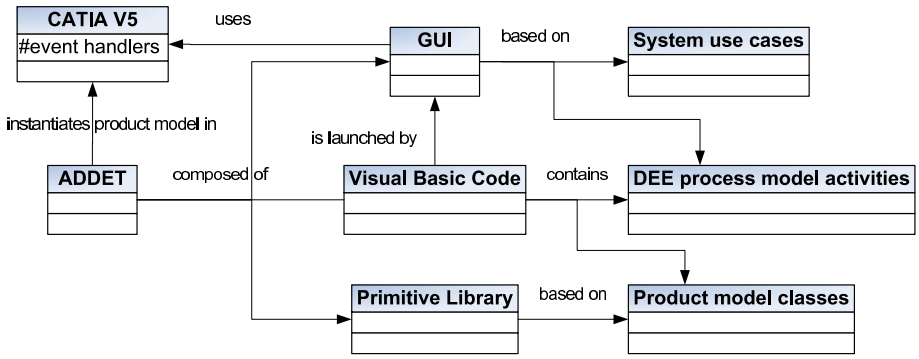


Figure A.1: Class diagram of the KBE application ADDET.

A.1 Product model classes as primitives

Figure A.2 shows the product model classes. For each of these classes a primitive has been programmed as a UDF. Another option for instantiating the product entities is to model each entity as separate geometry, attribute and operation. For instance, instantiating a zone entity would first require the definition of the surface based on four edges. Next parameters and rules are defined separately, parameters to function as zone attributes and rules for the behaviour of the zone. Finally, these parameters and rules should be linked to the surface element, creating the desired zone entity. The advantage of UDF's is that attributes and operations are be assigned to a single entity within CATIA.

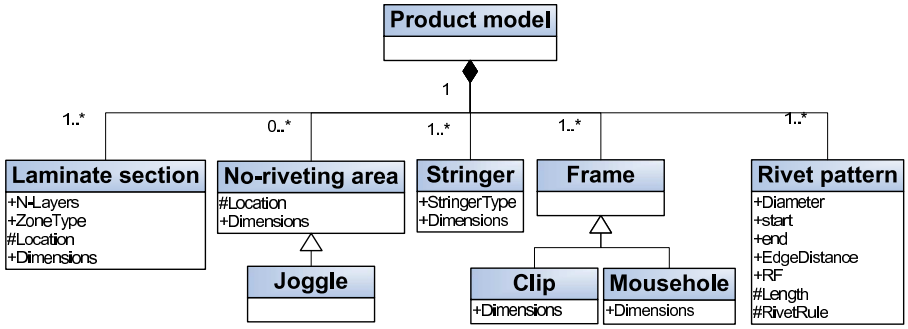


Figure A.2: Product model classes.

Figure A.3 shows the primitive defined for the riveting pattern. The geometric input information is the plane defining the frame location, the start of the pattern (point A), the end (point B), and the panel loft. The input dimensions are the rivet diameter, edge distance and minimum reserve factor (RF). Using this input information, the rivet primitive is instantiated in CATIA as shown in figure A.3, where the attributes are stored as children of the entity. The behaviour of the primitive is not shown, but is implicitly programmed in the primitive. For instance, if changing the rivet distance would result in violation of the rivet rules, the colour of the rivets will change from black to red.

A.2 Design process model steps as implemented in the code

This section will discuss the different steps in the process model as implemented in AD-DET.

Input generator

The required input information is stored in an excel sheet. Different sheets contain infor-

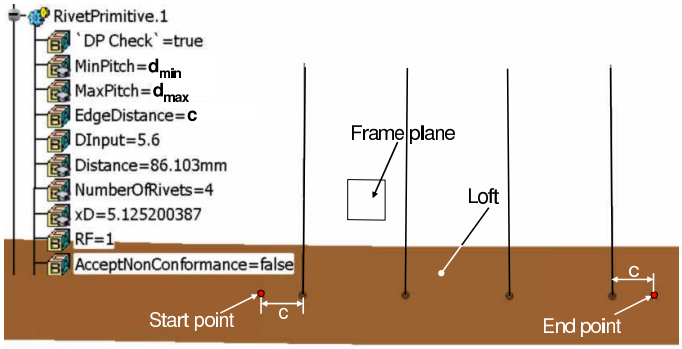


Figure A.3: Rivet primitive input geometry and attributes.

mation on requirements set for the design solution, laminate built-up, back-up structure information and design rules. A zone-wise laminate description is presented, were initial location, number of layers and zone-type are given. The initial location is defined by specifying the zone variables and assigning a value and reference frame or stringer datum (see section 4.1.1). Different sheets present the back-up information in terms of stringer dimensions, frame type (mouseholes or clips), and required rivet diameter. A sheet discusses the constants in the riveting rules for the different no-riveting areas constituted by joggles and splice areas. Since these constants may change for different projects, it is decided to give them as an input rather than hard-code them. Finally a sheet is used to specify at which frame stations the riveting pattern should be analysed.

For the first design loop the software user, in general a design engineer, will generate the input. For further cycles, the user can choose to automate the generation of the input, based on output from the search engine. Besides an excel input file, a CATIA model containing loft, frame and stringer information is provided by the user.

Product model generator

Based on the information specified in the input file and CATIA input model, an instantiation of the panel is generated in CATIA. First the zone edges are modeled as lines on the panel loft, based on the initial values and reference frame/stringer datum for the zone variables. Next the laminate is instantiated using the zone primitive (step 1, figure A.4). Input for the zone primitive are its four edges, the number of layers and the zone-type. Next the back-up structure is instantiated using the frame and stringer primitives. The start and end of the frame/stringer intersection is indicated using a point, and a representation of the intersection is instantiated using a frame primitive (step 2, figure A.5). Based on the design rules and accompanying constants specified in the input file, the

no-riveting areas are indicated by instantiating a start and end point at frame stations specified in the input file (step 3, figure A.6). Finally the rivet primitives are instantiated between the points of the frame/stringer intersections and the points of the no-riveting areas (step 4, figure A.7).

A report file is created by the product generator, containing a view on the product consisting of variable and riveting information. Besides the attribute values of the rivet primitives, information on its relation with zone variables is exported. Finally constraints on the zone variables are exported.

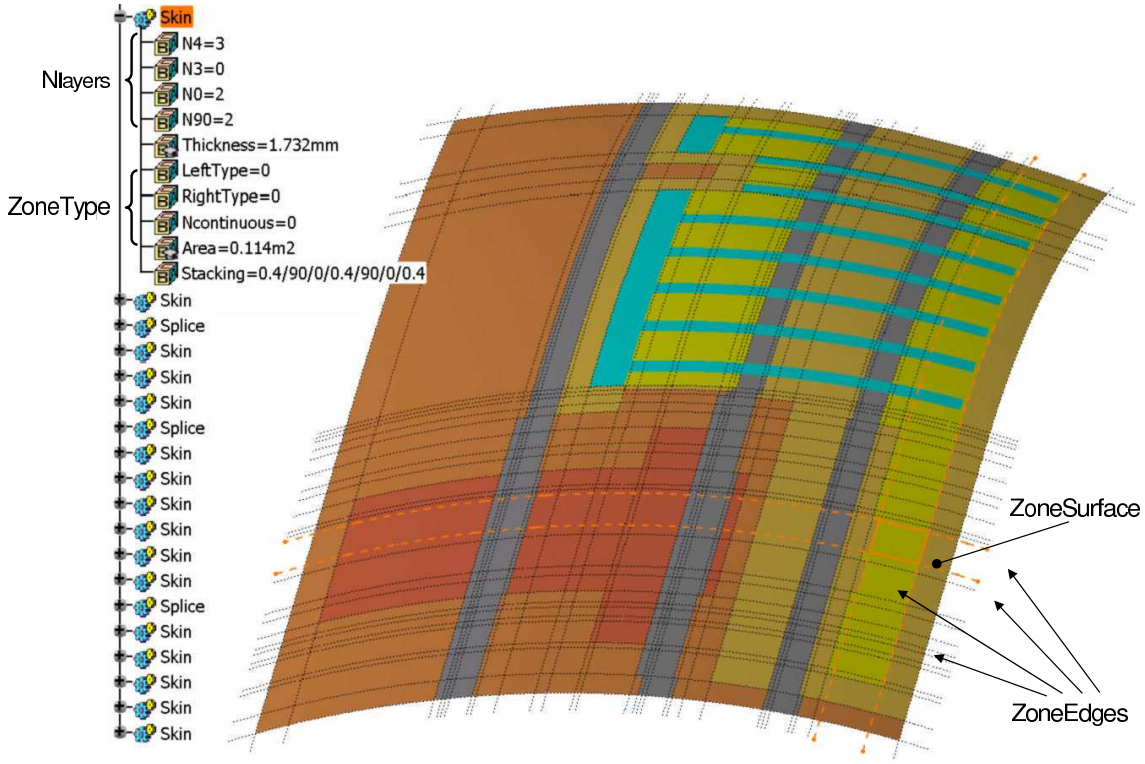


Figure A.4: Step 1: Instantiation of the zone primitives.

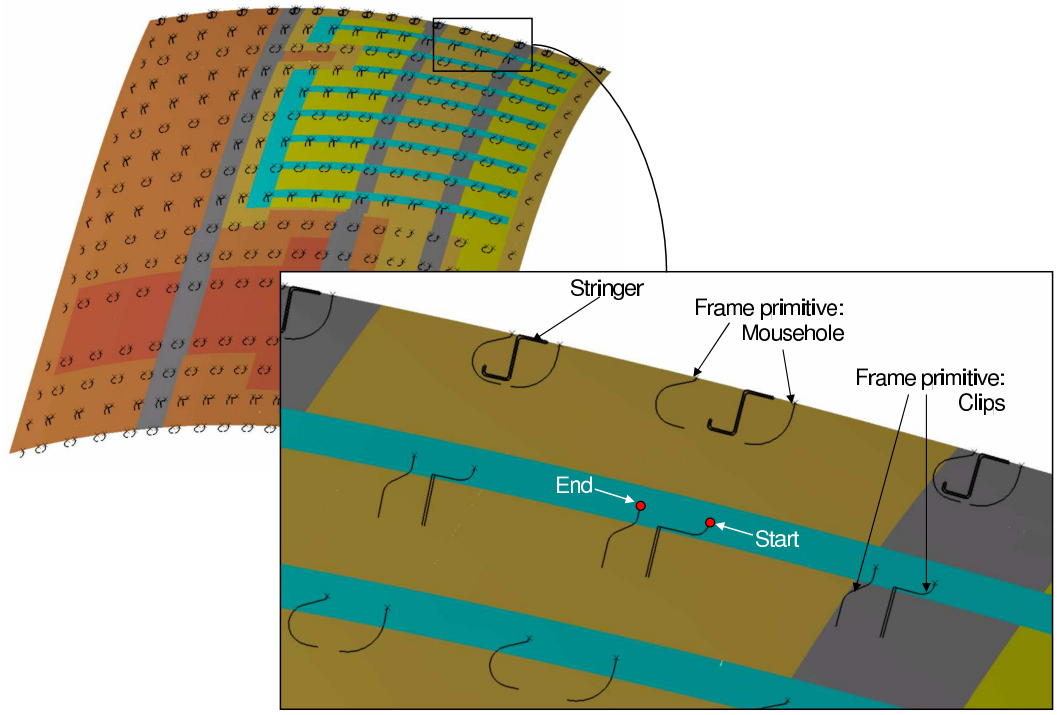


Figure A.5: Step 2: Instantiation of the back-up structure primitives.

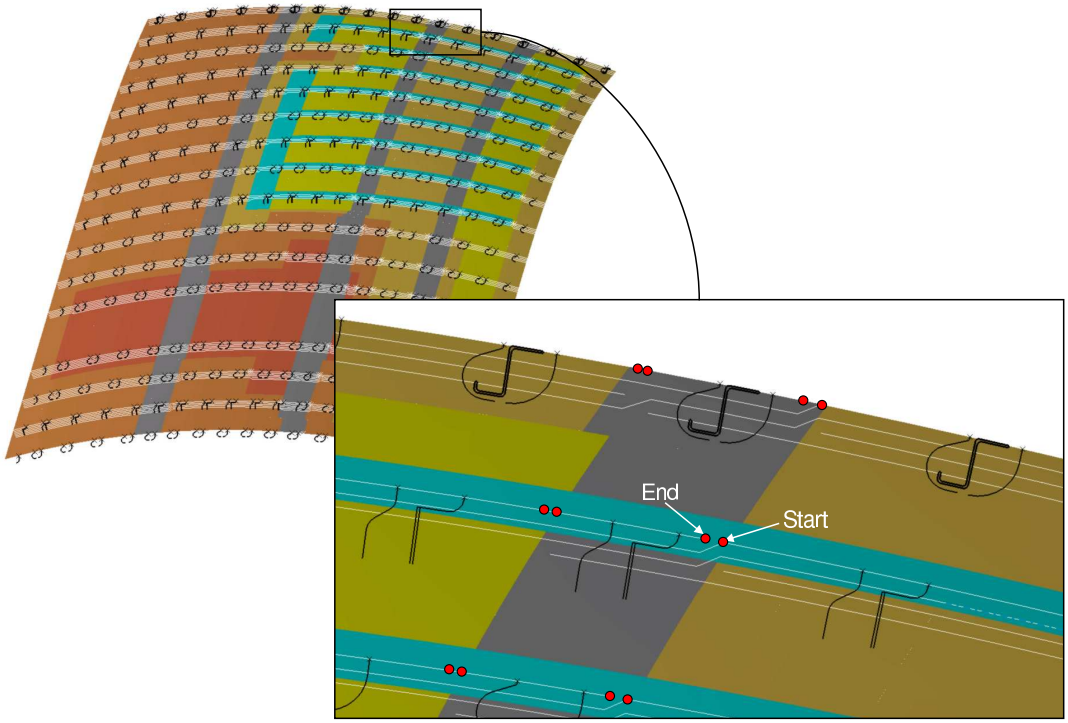


Figure A.6: Step 3: Instantiation of the no-riveting areas.

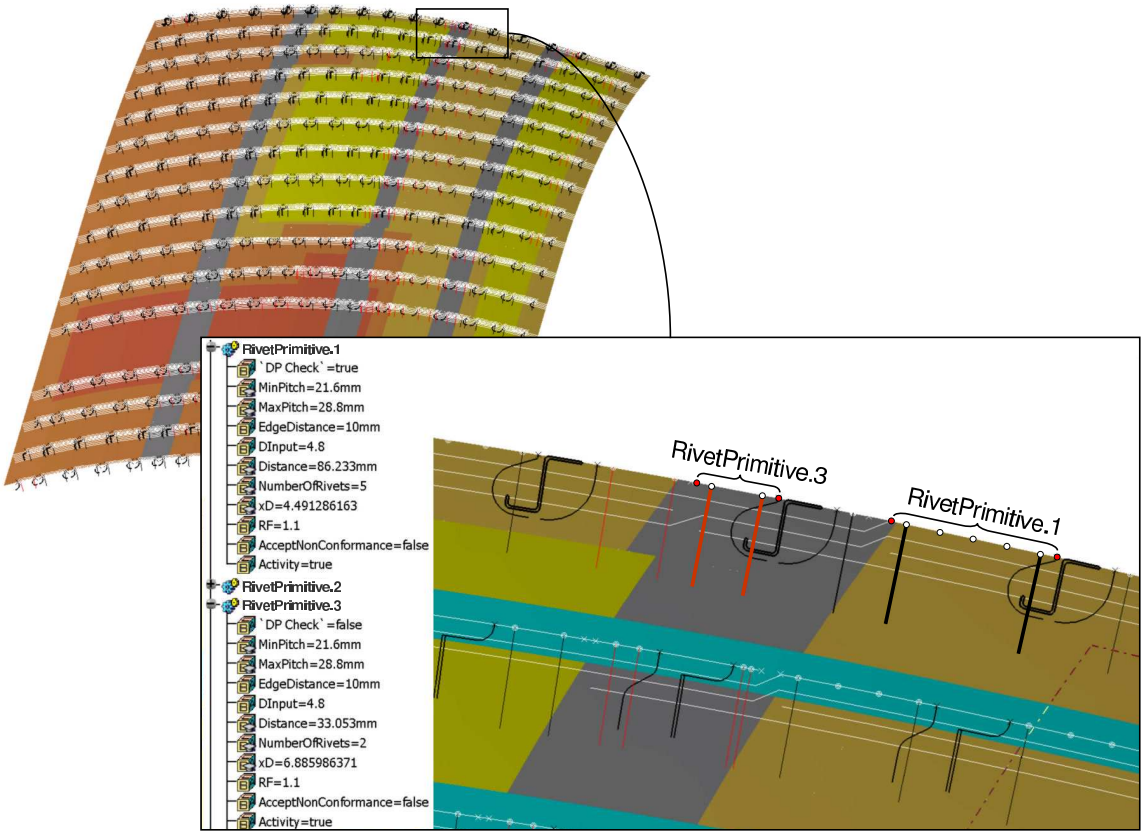


Figure A.7: Step 4: Instantiation of the rivet primitives.

Product analysis tool

The analysis tool uses an excel sheet to import the report file, which is exported by the product model generator. The state of the product in terms of constraint satisfaction and objective function value is determined. As discussed in chapter 4, the objective function value is a function of the sum of the cost of constraint violation. The cost of constraint violation is described using a penalty function, see chapter 5, and the analysis tool uses these functions to calculate the objective function value.

A file containing the information specified in the report file, and the state of the product is exported in a product property file, which is an excel file, by the analysis tool.

Evaluator

The evaluator compares the state of the product, as specified in the product property file, with the initial requirements set. For this design problem the requirement is that no hard constraints are violated. If this condition is not met, the product property file will be transferred to the search engine.

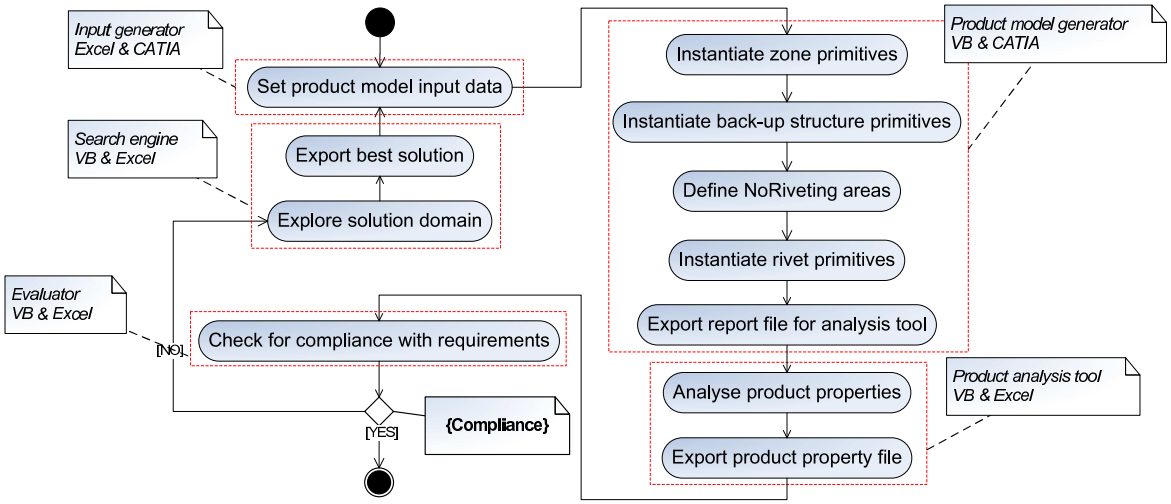


Figure A.8: Process model of the KBE application ADDET.

Search engine

The information in the product property file is used by the search engine to explore the solution domain and to find solutions to the design problem. One sheet in the product property file contains a list of zone variables and their values. A different sheet contains the rivet primitive data and the penalty functions to calculate the cost of constraint violation. Since information on the relation between zone variables and rivet primitive data is present in the product property file, changing a variable value will automatically update the rivet data and the corresponding value of the objective function.

The procedure of finding an optimum for the objective function is discussed in chapter 4. A list of variable values that result in a best solution is exported by the search engine, and put at the disposal of the input generator.

A.3 General software architecture

ADDET contains a GUI, which is used by the user to initiate different events. An event is an action or occurrence detected by the software[1], for instance the `Click-On(OpenInputFile)` event. The events uses different event-handlers to launch the actual code in the core, see figure A.9.

The general software architecture consists of a Graphical User Interface (GUI), event-handlers, a core containing the actual VB software code, and a repository of the primitives. The GUI of ADDET is created inside CATIA, using icons that are linked to event-handlers defined internally in CATIA. This information is stored in the CATSettings of CATIA.

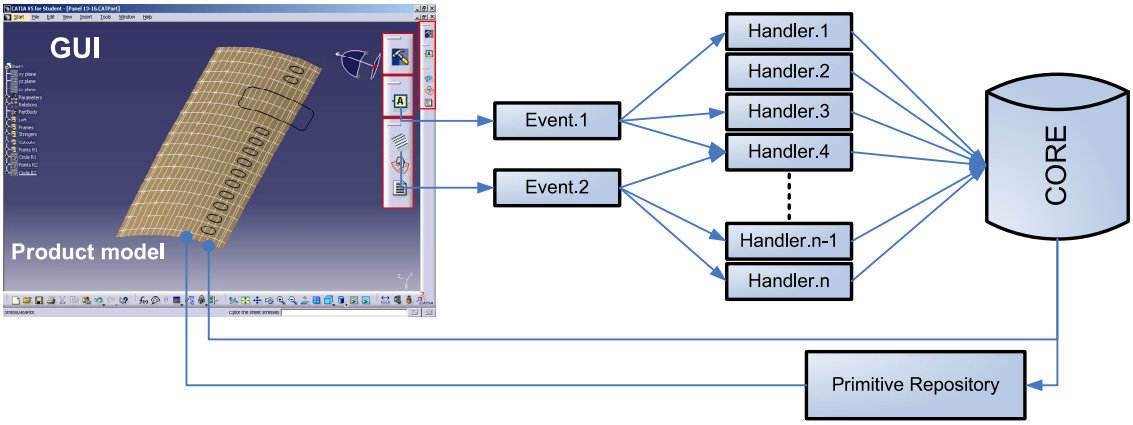


Figure A.9: Software architecture of ADDET.

The VB code in the core can be used to instantiate classes defined internally within CATIA V5, using the application programming interface (API). The VB code can also instantiate a primitive from the repository in CATIA.

Bibliography

[1] Palmer G.: Java event handling. *(Prentice Hall PTR, Upper Saddle River, USA, 2002)*

Appendix B

Efficiency and Effectiveness of the Solution Algorithm

In order to evaluate the efficiency and effectiveness of the heuristic solution algorithm, four different test problems are defined. Efficiency is evaluated in terms of the complexity reduction of the solution finding process, effectiveness is evaluated in terms of the number of solutions found. For each problem the number of structural entities is presented. The number of design variables is discussed by means of a top-view of the panel, showing changes in laminate lay-up. Only variables in circumferential direction are evaluated in the test problems. Next the number of constraints on rivet positions (type h1), and the number of constraints on zone dimensions (type h2) are given.

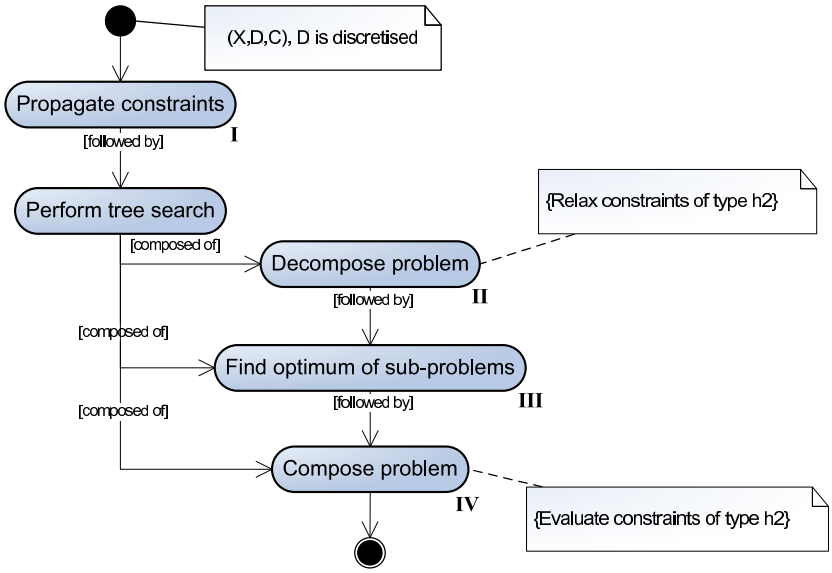


Figure B.1: Activity diagram showing the different steps in the solution finding algorithm.

Figure B.1 shows the different inference steps as programmed in the solution finding algorithm. The input for the algorithm is the design problem $\mathbf{P} = (\mathbf{X}, \mathbf{D}, \mathbf{C})$, where the variable domains D_i have been discretised. First step, step I, is to propagate the rivet constraints, reducing the discretised variable domains. The next step is the tree seacrh, which is composed of step II to step IV. Step II is to decompose the problem by relaxing the zone variables and grouping the design variables in clusters. Next the optimum for the decomposed problems is calculated in step III, and finally in step IV the problem is composed by evaluating the zone constraints. Based on these steps, the complexity of the algorithm can be evaluated, by summing the number of evaluations for step I to IV. Finally a worst-case scenario for the problem complexity is calculated to assess the efficiency of the solution finding algorithm. This worst case-scenario follows the same steps, but without constraint propagation and decompositioning of the problem. Eq-B.1 can be used to calculate the worst-case complexity of the solution finding process.

$$N_{evaluation} = \prod_{i=1}^{|\mathbf{X}|} |D_i| \cdot N_C \tag{B.1}$$

The steps in the solution finding algorithm, and the manner in which the accompanying complexity can be calculated, will be illustrated using a sample problem, shown in figure B.2.

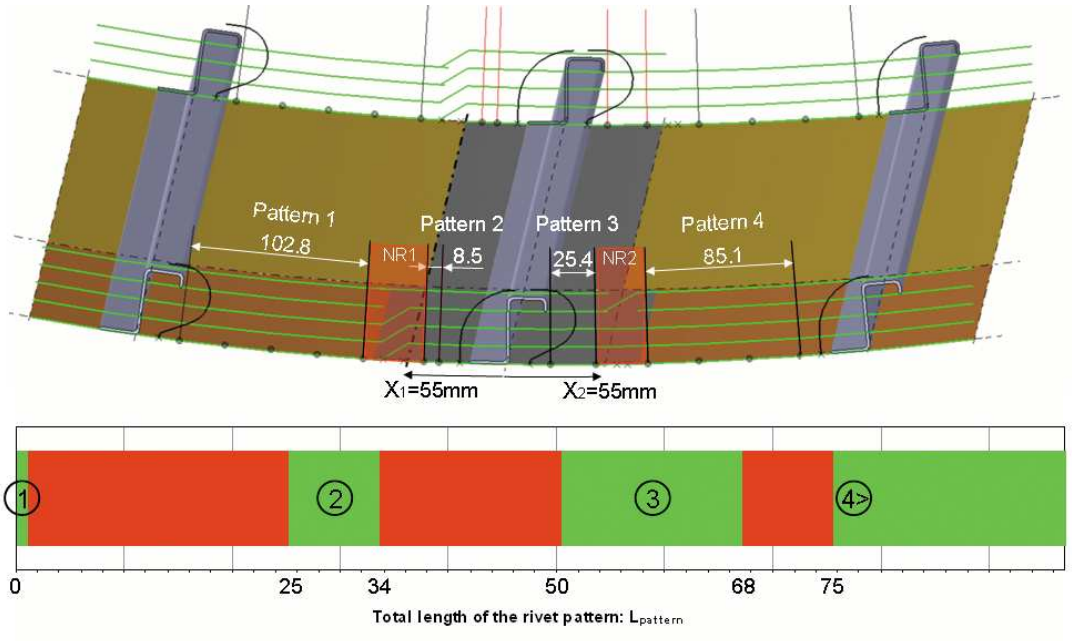


Figure B.2: Sample design problem, having two design variables, i.e., X_1 and X_2.

The problem has two design variables, i.e., X_1 and X_2, determining the location of the splice area in the laminate. The green lines give a schematic representation of the aluminium layers in the laminate, the black dots indicate a rivet position. Variable X_1 defines the location of a no-riveting area NR1 and influences the length of rivet patterns 1 and 2. Variable X_2 defines the location of a no-riveting area NR2 and influences the length of patterns 3 and 4. The relations are shown in the N^2 diagram in figure B.3.

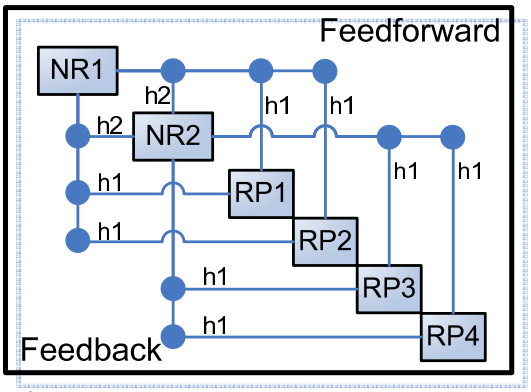


Figure B.3: N^2 diagram of the sample problem, showing the constraint relations.

The domains for the variables X_1 and X_2 are constituted by setting the length of the related rivet patterns equal to the values indicated in the lower part of figure B.2: {0, 25, 34, 50, 68, 75}. These values correspond to the rivet pattern lengths where the rivet requirements are fulfilled. This way, the problem becomes locally consistent with constraints type h1. Table B.1 shows the values that the two variables must have in order to comply with riveting constraint h1.

Table B.1: Constituting the discretised variable domains

$L_{pattern1}$	X_1	$L_{pattern2}$	X_1	$L_{pattern3}$	X_2	$L_{pattern4}$	X_2
0	157.8	0	46.5	0	29.6	0	140.1
25	132.8	25	71.5	25	54.6	25	115.1
34	123.8	34	80.5	34	63.6	34	106.1
50	107.8	50	96.5	50	79.6	50	90.1
68	89.8	68	114.5	68	97.6	68	72.1
75	82.8	75	121.5	75	104.6	75	65.1

Step I

When applying constraint propagation to the discretised variable domains, each variable value should be evaluated with respect to the constraints (type h1) constituted by the rivet patterns. The number of constraints of type h1 is equal to the number of rivet patterns

in the test problem. Therefore the number of evaluation for constraint propagation can be calculated as follows:

$$N_{propagation} = \sum_{i=1}^{|\mathbf{X}|} \mid D_i \mid \cdot N_{pattern} \tag{B.2}$$

For the sample problem eq-B.2 becomes $12 \cdot 8 + 12 \cdot 8 = 192$

Step II

The problem is decomposed by clustering the design variables. The cluster number is an attribute of the design variables, and the appropriate cluster number is assigned to the variables based on the decompositioning rules that are defined in the knowledge base. This step doesn't add to the solution finding complexity.

For the sample problem, X_1 and X_2 belong to different clusters, since they do not influence the same rivet pattern and therefor need not be varied simultaneously.

Step III

Having decomposed the problem, an optimum value for the objective function is determined for each sub-problem. This is done by evaluating all possible sets of variable values (the tree search). Eq-B.3 shows the resulting number of evaluations.

$$N_{optimise} = \sum_{i=1}^{N_{cluster}} dim \left[D_1 \times D_2 \times \cdots \times D_n \right]_{n=1..N_{var.i}} \tag{B.3}$$

Table B.2 shows the values in the variable domains and the corresponding values of the objective function. The number of evaluation according to eq-B.3 becomes $12 + 12 = 24$.

Step IV

Having determined the combination of variable values that result in an optimum value for the objective function of the decomposed problem, the problem should be composed by activating the zone constraints (type 2). The figures of each test problem will show the design variables and the existing constraints of type h2. The number of evaluations to assess if the zone constraints have been fulfilled are calculated using eq-B.4, where i and j indicate the specific variables related by h2.

$$N_{compose} = \sum_{n=1}^{N_{h2}} \left[|D_i| \cdot |D_j| \right]_n \tag{B.4}$$

Table B.2: Finding an optimum for the objective function

X1	Score	X2	Score
46.5	0	29.6	3.003
71.5	0	140.1	3.003
96.5	0	79.6	3.005
107.8	0	90.1	3.005
132.8	0	115.1	3.006
157.8	0	54.6	3.008
80.5	0.102	63.6	3.105
123.8	0.102	106.1	3.105
82.8	*1.503*	*65.1*	*4.306*
121.5	*1.503*	*104.6*	*4.306*
89.8	*3.153*	*72.1*	*6.156*
114.5	*3.153*	*97.6*	*6.156*

For the sample problem, the variable domains have been reduced to 8 values after optimisation. Since the width of a splice is limited by a minimum, there is a constraint h2 acting on the two variables that needs to be evaluated. The number of evaluation is according to eq-B.4: $8 \cdot 8 = 64$.

Number of solutions

The problem is decomposed in several sub-problems by relaxing the constraint of type h2. The resulting sub-problems are described by a cluster of design variable, related by riveting constraint (type h1). After composing the problem and applying the zone constraints type h2, different non-related sub-problems can still be identified. The variables in these sub-problems have the same dimension of the solution vector, and are not related to variables in the other sub-problems. The sample problem is composed of only one problem, but the four test problems do have these sub-problems, as can be seen in tables B.8, B.14, B.20 and B.26.

Having composed the problem, the total number of solutions is calculated. Mathematically spoken, the number of solutions of the composed problem would be all possible variable value combinations that are a solution to the problem, i.e., a multiplication of the number of solutions for each variable. From an engineering point of view, however, combining all possible local solutions will not result in significantly different global solutions. Therefore, the number of solutions is given as the maximum number of solutions for the sub-problems after composing the problem.

B.1 Test problem P1

Test problem descriptions Figure B.4 shows a topview of the laminate, and indicates the design variables in circumferential direction and the zone constraints relating different variables. The number of structural entities is shown in table B.3.

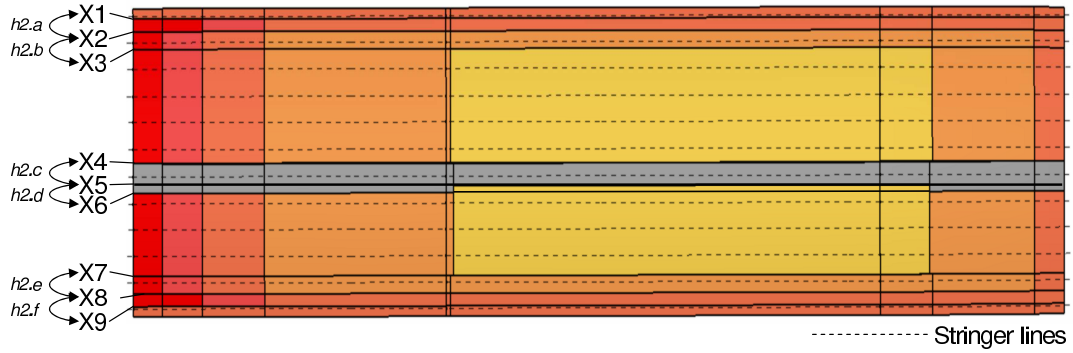


Figure B.4: Top-view of test problem P1.

Table B.3: Order of magnitude of the structural entities

Entity	RivetPatterns	Zones	Frames	Stringers	NoRivet Areas
$O[\#]$	123	54	16	12	20

Dimensions of the design variable domains Table B.4 shows the dimensions of the discretised variable domains before and after constraint propagation. Using eq-B.2 the number of evaluations is $1.2 \cdot 10^5$.

Table B.4: Dimensions of the design variable domains

Variable	X1	X2	X3	X4	X5	X6	X7	X8	X9	
$\mid \mathbf{D} \mid$	1	120	60	220	280	120	60	120	1	*discretised*
	1	31	16	35	1	40	16	31	1	*propagated*

Find optimum of decomposed problem The problem is decomposed by clustering the design variables, see table B.5. According to eq-B.3, the number of evaluations is 679.

Compose the problem Table B.6 shows the variable domain dimensions after optimising the decomposed problem. Figure B.4 shows the design variables and the existing constraints of type h2. The number of evaluations to assess if the zone constraints have been fulfilled are calculated as follows:

Table B.5: Decompose the problem by clustering of the design variables

Variable	X1	X2	X3	X4	X5	X6	X7	X8	X9
Cluster	I	I	II	III	IV	IV	V	VI	VI

$$N_{compose} = \sum_{n=1}^{6} \left[|D_i| \cdot |D_j|\right]_n = (|X_1|) + (|X_1| \cdot |X_3|) + \cdots$$
$$+(|X_4| \cdot |X_5|) + (|X_4| \cdot |X_5|) + (|X_7| \cdot |X_8|) + (|X_7| \cdot |X_8|) = 1.7 \cdot 10^3$$

Table B.6: Dimensions of the optimum design variable domains

Variable	X1	X2	X3	X4	X5	X6	X7	X8	X9
$\mid \mathbf{D} \mid$	31	31	16	13	7	7	16	31	31

Calculate the solution finding complexity The complexity of the solution finding algorithm, expressed in number of variable value evaluations, is calculated by summing the number of evaluations of the previous steps:

$$N_{evaluation} = N_{propagation} + N_{optimise} + N_{compose} = 1.2 \cdot 10^5$$

$$N_{evaluation} = 4.7 \cdot 10^{16} \quad \text{Worst-Case}$$

Number of solutions Table B.7 shows the number of solutions for the different sub-problems. After composing the problem and applying the zone constraints (type 2), the number of possible values for each variable is given, and the total number of solutions can be assessed as the maximum number of solutions for the sub-problems (bold number in table B.8).

Table B.7: Number of solutions for the decomposed problem

Variable	X1	X2	X3	X4	X5	X6	X7	X8	X9
Decomposed	1	8	6	5	3	3	6	7	7

Table B.8: Number of solutions for the composed problem

Variable	X1	X2	X3	X4	X5	X6	X7	X8	X9
Composed	**7**	**7**	**7**	3	3	3	5	5	5

$$N_{solution} = 7$$

B.2 Test problem P2

Test problem descriptions Figure B.5 shows a topview of the laminate, and indicates the design variables in circumferential direction and the zone constraints relating different variables. The number of structural entities is shown in table B.9.

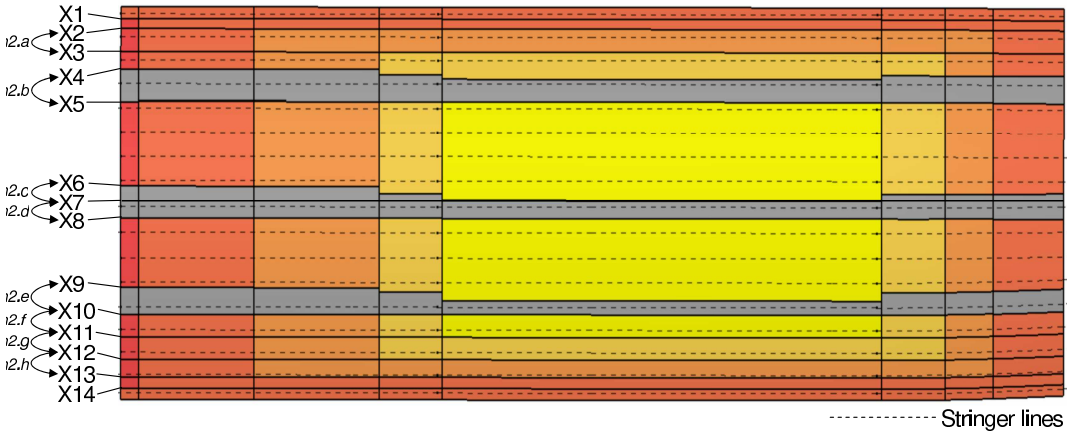


Figure B.5: Top-view of test problem P2.

Table B.9: Order of magnitude of the structural entities

Entity	RivetPatterns	Zones	Frames	Stringers	NoRivet Areas
$O[\#]$	207	104	16	17	34

Dimensions of the design variable domains Table B.10 shows the dimensions of the discretised variable domains before and after constraint propagation. Using eq-B.2 the number of evaluations is $5.1 \cdot 10^5$.

Table B.10: Dimensions of the design variable domains

Variable	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	X11	X12	X13	X14	
| **D** |	100	100	60	280	220	60	300	220	60	340	340	100	100	180	*discretised*
	15	5	2	4	11	3	8	24	3	13	14	9	8	4	*propagated*

Find optimum of decomposed problem The problem is decomposed by clustering the design variables, see table B.11. According to eq-B.3, the number of evaluations is 138.

Compose the problem Table B.12 shows the variable domain dimensions after optimising the decomposed problem. Figure B.5 shows the design variables and the existing

Table B.11: Decompose the problem by clustering of the design variables

Variable	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	X11	X12	X13	X14
Cluster	I	II	III	III	IV	V	V	VI	VII	VIII	IX	X	XI	XII

constraints of type h2. The number of evaluations to assess if the zone constraints have been fulfilled are calculated as follows:

$$N_{compose} = \sum_{n=1}^{8} [|D_i| \cdot |D_j|]_n = (|X_2| \cdot |X_3|) + (|X_2| \cdot |X_3| \cdot |X_5|) + (|X_6|) + \cdots$$
$$+(|X_6| \cdot |X_8|) + (|X_9| \cdot |X_{10}|) + (|X_9| \cdot |X_{10}| \cdot |X_{11}|) + \cdots$$
$$+(|X_9| \cdot |X_{10}| \cdot |X_{11}| \cdot |X_{12}|) + (|X_9| \cdot |X_{10}| \cdot |X_{11}| \cdot |X_{12}| \cdot |X_{13}|) = 5.8 \cdot 10^5$$

Table B.12: Dimensions of the optimum design variable domains

Variable	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	X11	X12	X13	X14
$\mid \mathbf{D} \mid$	23	23	6	6	12	6	6	19	16	11	12	16	16	2

Calculate the test problem complexity The complexity of the solution finding algorithm, expressed in number of variable value evaluations, is calculated by summing the number of evaluations of the previous steps:

$$N_{evaluation} = N_{propagation} + N_{optimise} + N_{compose} = 1.1 \cdot 10^6$$

$$N_{evaluation} = 3.8 \cdot 10^{32} \quad \text{Worst-Case}$$

Number of solutions Table B.13 shows the number of solutions for the different sub-problems. After composing the problem and applying the zone constraints (type 2), the number of possible values for each variable is given, and the total number of solutions can be assessed as the maximum number of solutions for the sub-problems (bold number in table B.14).

Table B.13: Number of solutions for the decomposed problem

Variable	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	X11	X12	X13	X14
Cluster	5	4	3	3	5	3	3	5	4	3	5	4	4	2

$$N_{solution} = 13$$

Table B.14: Number of solutions for the composed problem

Variable	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	X11	X12	X13	X14
Cluster	7	7	5	5	5	**13**	**13**	**13**	4	4	4	4	4	4

B.3 Test problem P3

Test problem descriptions Figure B.6 shows a topview of the laminate, and indicates the design variables in circumferential direction and the zone constraints relating different variables. The number of structural entities is shown in table B.15.

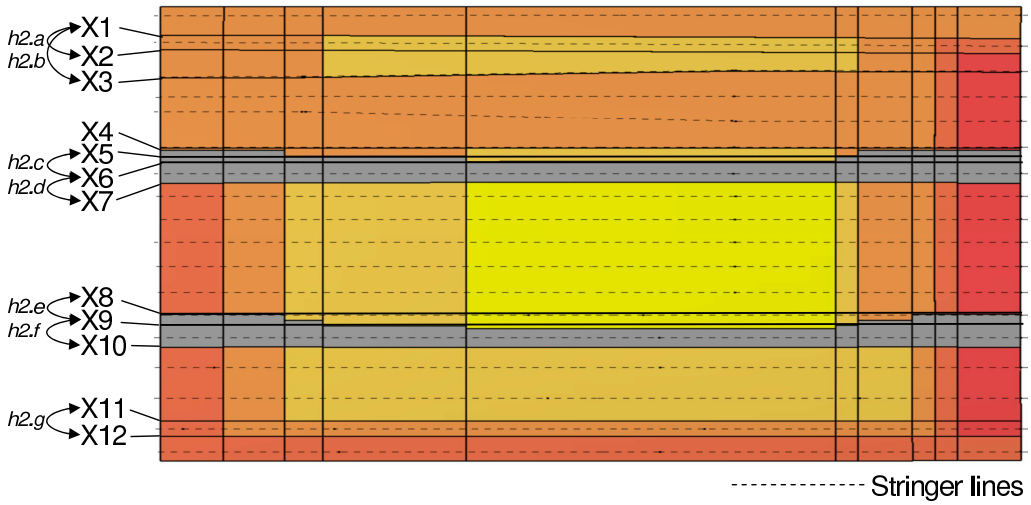


Figure B.6: Top-view of test problem P3.

Table B.15: Order of magnitude of the structural entities

Entity	RivetPatterns	Zones	Frames	Stringers	NoRivet Areas
$O[\#]$	173	110	7	18	31

Dimensions of the design variable domains Table B.16 shows the dimensions of the discretised variable domains before and after constraint propagation. Using eq-B.2 the number of evaluations is $3.6 \cdot 10^5$.

Find optimum of decomposed problem The problem is decomposed by clustering the design variables, see table B.17. According to eq-B.3, the number of evaluations is 414.

Table B.16: Dimensions of the design variable domains

Variable	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	X11	X12	
| **D** |	180	60	1	1	120	300	280	60	300	280	220	280	*discretised*
	16	15	1	1	7	14	33	16	10	44	46	1	*propagated*

Table B.17: Decompose the problem by clustering of the design variables

Variable	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	X11	X12
Cluster	I	II	III	IV	IV	IV	V	VI	VI	VII	VIII	IX

Compose the problem Table B.18 shows the variable domain dimensions after optimising the decomposed problem. Figure B.6 shows the design variables and the existing constraints of type h2. The number of evaluations to assess if the zone constraints have been fulfilled are calculated as follows:

$$N_{compose} = \sum_{n=1}^{7} \left[|D_i| \cdot |D_j|\right]_n = (|X_1| \cdot |X_2|) + (|X_1| \cdot |X_2| \cdot |X_3|) + \cdots$$
$$+(|X_6|) + (|X_6| \cdot |X_7|) + (|X_9|) + (|X_9| \cdot |X_{10}|) + (|X_{11}| \cdot |X_{12}|) = 6.5 \cdot 10^3$$

Table B.18: Dimensions of the optimum design variable domains

Variable	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	X11	X12
| **D** |	16	15	1	96	96	96	33	61	61	44	46	1

Calculate the test problem complexity The complexity of the solution finding algorithm, expressed in number of variable value evaluations, is calculated by summing the number of evaluations of the previous steps:

$$N_{evaluation} = N_{propagation} + N_{optimise} + N_{compose} = 3.7 \cdot 10^5$$

$$N_{evaluation} = 5.8 \cdot 10^{24} \quad \text{Worst-Case}$$

Number of solutions Table B.19 shows the number of solutions for the different sub-problems. After composing the problem and applying the zone constraints (type 2), the number of possible values for each variable is given, and the total number of solutions can be assessed as the maximum number of solutions for the sub-problems (bold number in table B.20).

$$N_{solution} = 24$$

Table B.19: Number of solutions for the decomposed problem

Variable	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	X11	X12
Decomposed	2	4	1	6	6	6	3	8	8	5	5	1

Table B.20: Number of solutions for the composed problem

Variable	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	X11	X12
Composed	10	10	10	9	9	9	9	**24**	**24**	**24**	7	7

B.4 Test problem P4

Test problem descriptions Figure B.7 shows a topview of the laminate, and indicates the design variables in circumferential direction and the zone constraints relating different variables. The number of structural entities is shown in table B.21.

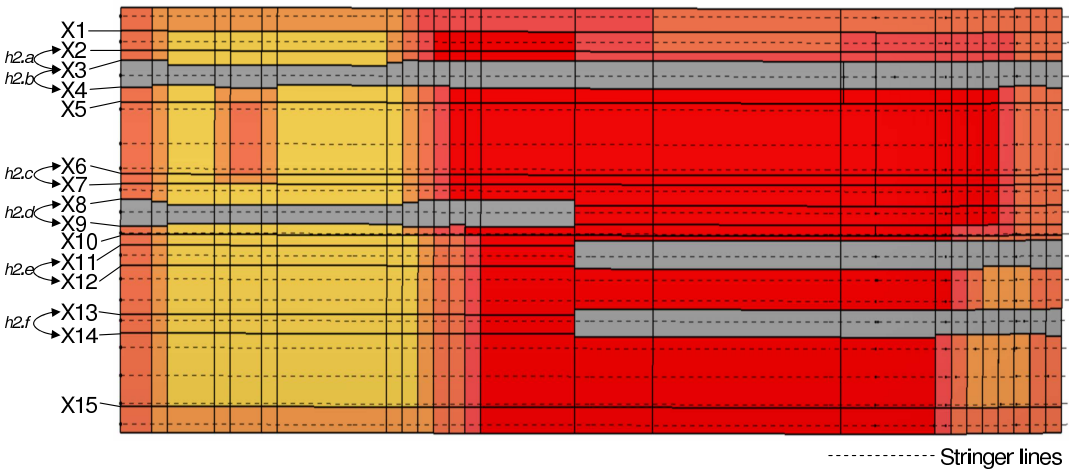


Figure B.7: Top-view of test problem P4.

Table B.21: Order of magnitude of the structural entities

Entity	RivetPatterns	Zones	Frames	Stringers	NoRivet Areas
$O[\#]$	239	214	16	18	33

Dimensions of the design variable domains Table B.22 shows the dimensions of the discretised variable domains before and after constraint propagation. Using eq-B.2 the number of evaluations is $3.7 \cdot 10^5$.

Table B.22: Dimensions of the design variable domains

Variable	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	X11	X12	X13	X14	X15	
| **D** |	210	120	60	340	240	60	60	60	1	60	60	220	1	1	60	*discretised*
	5	17	14	1	19	2	17	24	1	8	6	4	1	1	14	*propagated*

Find optimum of decomposed problem The problem is decomposed by clustering the design variables, see table B.23. According to eq-B.3, the number of evaluations is 724.

Table B.23: Decompose the problem by clustering of the design variables

Variable	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	X11	X12	X13	X14	X15
Cluster	I	II	II	II	III	III	IV	IV	V	VI	VII	VIII	IX	X	XI

Compose the problem Table B.24 shows the variable domain dimensions after optimising the decomposed problem. Figure B.7 shows the design variables and the existing constraints of type h2. The number of evaluations to assess if the zone constraints have been fulfilled are calculated as follows:

$$N_{compose} = \sum_{n=1}^{6} \left[|D_i| \cdot |D_j|\right]_n = (|X_3|) + (|X_4|) + \cdots$$
$$+(|X_6| \cdot |X_7|) + (|X_6| \cdot |X_7| \cdot |X_9|) + (|X_{11}| \cdot |X_{12}|) + (|X_{13}| \cdot |X_{14}|) = 1.3 \cdot 10^3$$

Table B.24: Dimensions of the optimum design variable domains

Variable	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	X11	X12	X13	X14	X15
| **D** |	5	16	16	16	14	14	46	46	1	8	6	4	1	1	14

Calculate the test problem complexity The complexity of the solution finding algorithm, expressed in number of variable value evaluations, is calculated by summing the number of evaluations of the previous steps:

$$N_{evaluation} = N_{propagation} + N_{optimise} + N_{compose} = 3.8 \cdot 10^5$$

$$N_{evaluation} = 3.2 \cdot 10^{26} \quad \text{Worst-Case}$$

Number of solutions Table B.25 shows the number of solutions for the different subproblems. After composing the problem and applying the zone constraints (type 2), the

number of possible values for each variable is given, and the total number of solutions can be assessed as the maximum number of solutions for the sub-problems (bold number in table B.26).

Table B.25: Number of solutions for the decomposed problem

Variable	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	X11	X12	X13	X14	X15
Decomposed	3	12	12	12	6	6	9	9	1	3	1	2	1	1	5

Table B.26: Number of solutions for the composed problem

Variable	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	X11	X12	X13	X14	X15
Composed	3	1	1	1	**10**	**10**	**10**	**10**	**10**	3	1	1	1	1	5

$$N_{solution} = 10$$

Samenvatting

KENNIS GEBASEERDE METHODE VOOR HET OPLOSSEN VAN COMPLEXITEIT IN TECHNISCHE PROBLEMEN

door Brent Vermeulen

Het ontwerpen van vliegtuig systemen wordt steeds complexer, vanwege het toenemende aantal eisen waaraan voldaan dient te worden. Daarbij komt dat de kennis om dergelijk complexe systemen te ontwerpen steeds minder beschikbaar is voor bedrijven. Dit vanwege een afnemende beschikbaarheid van ingenieurs in de westerse wereld, en een reductie van het aantal mogelijkheden om kennis te kunnen overdragen. Bedrijven zullen zich moeten richten op het te gelde maken van de intern aanwezige kennis, om succesvol aan bovenstaande uitdagingen het hoofd te kunnen bieden.

Het in dit rapport gepresenteerde onderzoek zal bijdragen aan het oplossen van bovenstaande uitdagingen. Een kennis gebaseerde methode voor het oplossen van complexiteit in gedetailleerde ontwerpproblemen wordt gepresenteerd. Aan de hand van een gedetailleerd probleem in het ontwerpproces van vezelmetaal laminaat (VML) romppanelen, zal het proces van het opzetten van een oplossingsstrategie wordt behandeld. De principes van knowledge based engineering (KBE) worden gebruikt om een software systeem te ontwerpen, dat oplossingen voor het ontwerpprobleem genereerd.

Gezien het creatieve karakter van een ontwerpproces zullen iteraties, en daarmee wijzigingen in het ontwerp, onvermijdelijk zijn. Het doorgeven van wijzigingen tussen disciplines wordt bemoeilijkt door het feit dat disciplines verschillende modellen van het product gebruiken voor hun analyses, resulterend in het herdefiniëren van de informatie. Kijkend naar het detail ontwerpproces van VML romppanelen, komt daar bij dat het definiëren van een te realiseren product in zich zelf sterk iteratief is. Gedetailleerde informatie over hoe het laminaat is bevestigd aan het geraamte waar de romp is nodig, en de positie van elke klinknagel dient hierbij te worden gecheckt.

Op het moment dat de kennis van de expert over hoe het ontwerpproces uit te voeren is geformaliseerd, kan deze kennis hergebruikt worden om het proces te automatiseren. Dit is het hoofdprincipe van KBE. KBE is de wetenschap van het identificeren, vastleggen en hergebruiken van ontwerp kennis. De focus bij het ontwikkelen van KBE applicaties ligt op het automatiseren van repetitieve en weinig creatieve ontwerpprocessen. De interpretatie van het ontwerpproces door verschillende experts is geformaliseerd, met behulp van een product en een proces model.

Een mathematisch model van het ontwerpprobleem, gebaseerd op de interpretatie van de expert, is gedefinieerd. In het mathematische model dient er tenminste één doelfunctie gedefinieerd te zijn, die beschreven wordt door een vector van ontwerp variabelen, wier domein beperkt worden door randvoorwaarden. De variabelen in het model bepalen de opbouw van het laminaat, de randvoorwaarden komen uit de eisen aan de laminaat opbouw en hoe het laminaat aan het geraamte van de romp verbonden wordt. Het mathematische probleem kan beschreven worden als een zogenaamd constraint satisfaction problem (CSP), waarbij een oplossing gevonden is als aan alle randvoorwaarden voldaan is. Soms is het toegestaan om niet aan een randvoorwaarde te voldoen; hier staat echter een penalty tegenover. De doelfunctie in het model is de som van alle penalties, welke geminimaliseerd dient te worden. Een heuristisch oplossingsalgoritme is toegesneden op het mathematische model, gebruikmakend van zowel de geformaliseerde expert kennis als van domein onafhankelijke oplossingsstrategieën.

Een software systeem, voor het oplossen van het detail ontwerpprobleem in VML romppanelen, genaamd ADDET, is ontwikkeld en geïmplementeerd in het ontwerpproces. De effectiviteit van ADDET met betrekking tot het genereren van oplossingen voor praktijk problemen is geëvalueerd. Behalve dat ADDET praktische oplossingen genereert, is er tevens een reductie in ontwerptijd van 60% geconstateerd. Vanwege deze reductie in ontwerptijd zullen ontwerp resultaten in een korter tijdbestek gegenereerd kunnen worden. Dit kan resulteren in knelpunten bij andere disciplines, bijvoorbeeld om dat zij de resultaten moeten checken in minder tijd. Een proceswijde optimalisatie is derhalve nodig, om te voorkomen dat de winst door automatisering verspeeld wordt door het ontstaan van dergelijke knelpunten.

Om de kennis van het ontwikkelen van een oplossingsstrategie te kunnen hergebruiken, is er een formele beschrijving van de methode gepresenteerd. Hierbij is gebruik gemaakt van een ontologie. Een ontologie is een expliciete beschrijving van de concepten en hun relaties in een bepaald domein, waarmee een formele vocabulaire gedefinieerd wordt.

Een probleem onafhankelijke oplossingsmethode voor technische problemen is ontwikkeld, welke probleem specifiek gemaakt wordt door de strategie van het zoeken naar oplossingen toe te snijden op het probleem voor handen. Een gestructureerde aanpak wordt voorgesteld, waarbij het probleem uit de praktijk, via een formele beschrijving van de interpretatie van de expert, omgeschreven wordt tot een mathematisch model. De oplossing strategie wordt vervolgens toegesneden op het mathematische model voor handen, door domein onafhankelijke oplossingsmethoden te combineren met probleemspecifieke kennis. Tenslotte dient het oplossingssysteem geïmplementeerd te worden in het ontwerpproces, om te kunnen verifiëren of er daadwerkelijk oplossingen voor het praktijk probleem gegenereerd worden.

About the Author

Brent Vermeulen

Born August the 20^{th}, 1976 in Woerden, The Netherlands

1988-1994	Gymnasium (pre-university education) at the Dr. F.H. de Bruyne Lyceum, Utrecht.
1994-2002	Aerospace Engineering at Delft University of Technology. Master project at the Production Technology chair concerning a feasibility analysis of an ellipsoid shaped airship, capable of lifting 200 tons of cargo. A preliminary design has been presented, including market analysis, aerodynamic, structural and stability analysis.
2002-2003	Benchmark study of aluminium alloys, comparing baseline Al2024 and Al7075 with CFRP and new advanced aluminium alloys. Executed in cooperation with Pechiney-Aerospace. This project included a literature study, design case study of a fuselage section and a market analysis.
2003-2006	PhD research at Delft University of Technology, under the supervision of prof. dr. ir. M.J.L. van Tooren. The research resulted in an Automated Detailed DEsign Tool (ADDET) for the automatic generation of detailed designs of Fibre Metal Laminate fuselage panels, using the principles of Knowledge Based Engineering.
2006-Now	Knowledge engineer at Stork Fokker AESP.