An increase in productivity is required for the European aviation industry to remain competitive in a world with rising competition, shorter development timescales, more complex products and decreasing numbers of technically skilled personnel. Fokker Elmo, a Dutch aircraft electrical wiring harness manufacturer, is facing these challenges. At present, preparing a wiring harness for manufacturing is a repetitive, time-consuming and mostly manual, experience-based process.

The research presented in this thesis aims at developing techniques to largely automate the generation of wiring harness manufacturing drawings, using Knowledge Based Engineering (KBE). A development approach is proposed as well as KBE building blocks. Three KBE applications are developed and tested that offer functionalities not present in general-purpose CAD systems. These applications can considerably reduce the amount of repetitive work, while ensuring compliance to physical constraints and manufacturing guidelines.
Harnessing the potential of Knowledge Based Engineering in manufacturing design

Tobie van den Berg
Harnessing the potential of Knowledge Based Engineering in manufacturing design

PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus prof. ir. K.C.A.M. Luyben,
voorzitter van het College voor Promoties,
in het openbaar te verdedigen op woensdag 18 december 2013 om 10:00 uur
door

Tobie VAN DEN BERG

ingenieur in de luchtvaart en ruimtevaart
egeboren te Nacala-Porto, Mozambique
Dit proefschrift is goedgekeurd door de promotor:

Prof.dr. ir. M.J.L. van Tooren

Samenstelling promotiecommissie:

Rector Magnificus voorzitter
Prof. dr. ir. M.J.L. van Tooren Technische Universiteit Delft, promotor
Prof. dr. ir. L.L.M. Veldhuis Technische Universiteit Delft
Prof. ir. J.J. Hopman Technische Universiteit Delft
Priv.-Doz.dr.-ing.habil. S. Rudolph Universität Stuttgart
Dr. J.S.M. Vergeest Technische Universiteit Delft
Dr. ir. G. La Rocca Technische Universiteit Delft
S. Taylor Fokker Elmo B.V.
Prof.dr. R. Curran Technische Universiteit Delft, reservelid

ISBN 978-94-6203-476-1

Copyright © 2013 by Tobie van den Berg.

All rights reserved. No part of the material protected by this copyright notice may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage and retrieval system, without prior permission of the author.

Cover photo © 2013 by Tobie van den Berg.

Printed by Wörmann Print Service, Zutphen, the Netherlands.
This research was made possible by the expertise of and close collaboration with Fokker Elmo and support from KE-works. This work presents results from the F3 'Formboard Fitting by Function' research project, a joint research project by Fokker Elmo and Delft University of Technology aiming at obtaining automatic formboard generation using Knowledge Based Engineering. The project was funded through a Dutch national Strategic Research Program by AgentschapNL and Fokker Elmo.
An increase in productivity is required for the European aviation industry to remain competitive in a world with more competition, shorter development time-scales, more complex products and decreasing numbers of technically skilled personnel. Fokker Elmo, a Dutch aircraft electrical wiring harness manufacturer, is facing these challenges. At present, preparing a wiring harness for manufacturing is a repetitive, time-consuming and mostly manual, experience-based process. An engineer must ensure that the wiring harness designed in 3D can be efficiently manufactured on a 2D production table, the formboard, such that the manufactured product can be readily installed in the 3D airplane. The associated customerspecific requirements are not accounted for by existing Computer Aided Design (CAD) systems, leading to numerous checks and model adjustments.

To increase the productivity of formboard design engineers and achieve consistent product quality, automation of the design process with Knowledge Based Engineering (KBE) and search methods is proposed. KBE systems can capture and systematically reuse product and process engineering knowledge in order to automate repetitive and non-creative design tasks. The iterative steps of a design process can be automated through the application of search or optimization techniques. This leads to the research proposition of this work:

**Research proposition.** The performance of the formboard design process can be increased in terms of time and quality by largely automating it using Knowledge Based Engineering.

The validity of the proposition is assessed by developing and evaluating KBE
solutions to support the formboard design process. This development requires answers to the research questions:

1. How to automate the formboard process steps?
2. How to model different geometric states of a product and their interdependencies in a KBE application?
3. What methodology can be used to develop a KBE solution that matches customer-specific requirements?

**Development approach**  
To partially or fully automate an engineering design process is a design activity by itself. Formal methods for the management, analysis and verification of complex systems are provided by the field of Systems Engineering (SE). The field of Knowledge Engineering (KE) gives methods to elicit and formalize domain-specific knowledge. Software development methods such as agile and incremental development approaches are provided by the field of Software Engineering (SWE). Elements from these fields and existing KBE development methodologies are combined into a heuristic development approach that is applied to the formboard design case. Iteratively, key processes and requirements are identified; knowledge is acquired, formalized, implemented in a KBE application and verified with domain experts. Prototype KBE application modules are used for elicitation and verification of acquired knowledge.

**KBE product model**  
KBE applications can be constructed using flexible parametric building blocks: High Level Primitive (HLP)s to generate product components and Capability Module (CM)s to extract discipline-specific views from these. The concept of the Structuring and Collating Node (SCN) is introduced in this work to specify classes that aggregate and define interdependencies between HLPs and CMs. Instead of developing a single KBE solution, it can be beneficial to divide the design process in several stages. A Multiple Domain Matrix (MDM) can be used to identify these stages, resulting in applications built with stage-specific HLPs. In the studied design case, the wiring product manufacturing system requires the wiring harness definition to be transformed from its 3D design state to a 2D manufacturing state. Multi-state HLPs are defined with geometry attributes corresponding to different states, which share parameters and can be interdependent.

**Analysis and flattening**  
The capabilities of current analysis and flattening tools are limited, requiring time-consuming, repetitive manual work. The automatic flattening operation provided by the electrical workbench of the available CAD system
is unreliable and requires additional checks and adjustments. Procedures are developed to automatically evaluate the 3D wiring harness state for violations of bending and twisting flexibility limits with respect to the 2D manufacturing state and to evaluate the presence of unflattenable break-outs. Flexibility evaluations are performed by means of experiment-based rules, validated by years of industrial practice at Fokker Elmo. The results are reported to the engineer and are the main input for flattening. A new method for wiring harness flattening is devised that aligns the 2D geometry states of the wiring harness HLPs with respect to each other. The analysis and flattening application returns 2D models with deformations within allowed limits and the minimum amount of bundle twisting.

**Fitting and configuring the formboard** A flattened wiring harness model is currently adjusted manually to fit within a standard formboard frame size. This is a search problem with multiple quality and manufacturing efficiency objectives such as minimize frame size, eliminate crossings and optimize for manufacturing ergonomics. To ensure that the fitted wiring harness state can be deformed to the 3D installation state, engineers currently have to perform time-consuming checks and adjustments. To eliminate the need for these, the flattened model is discretized into bendable sections for which bending limits are established. These limits fully constrain both an engineer during manual fitting and automatic fitting methods using search methods. From experimentation with different automatic fitting approaches the so-called alignment method, based on manual fitting heuristics, is selected and implemented. This method attempts to align branches of bundles with respect to each other according to a limited set of bending and flipping strategies. A formboard frame is automatically selected from a set of standard sizes, the layout is adjusted with respect to manufacturing ergonomics and production instructions are added.

**CAD exchange** The exchange of 3D CAD data is a known challenge in multidisciplinary, concurrent engineering environments involving multiple suppliers. To integrate the formboard design KBE solutions into the existing electrical wiring development process, an approach is proposed using an ‘interpreter’ KBE application that transforms the 3D harness model as extracted from the given CAD model, via STEP AP214 files to a HLP parameterization. This approach provides an adequate solution to integrating external, custom-made software engineering solutions into the conventional design and engineering process.

**The potential of applying KBE** To determine whether the KBE applications reduce lead time and ensure the quality of formboard drawings, a validation was per-
formed using 25 wiring harnesses from a commercial aircraft program. The quality of the output formboard drawings was approved by formboard design experts. The time spent on formboard design tasks was demonstrated to reduce from values in the order of hours to minutes. A conservative calculation indicates that, in practice, a five-fold productivity increase can be obtained.

The developed formboard KBE suite offers functionalities that are not present in conventional CAD systems. The KBE applications are geared to specific engineering processes that are (and will remain) outside the scope of general-purpose CAD systems. The applications can considerably reduce the amount of repetitive work, while ensuring compliance to physical constraints and manufacturing guidelines.
# Contents

Summary vii

Contents xi

Acronyms xv

## 1 Knowledge Based Engineering to support manufacturing design 1

1.1 Complexity of aircraft electrical wiring manufacturing design ..... 2
1.2 Knowledge Based Engineering & search ..... 6
1.3 KBE supported formboard design challenges ..... 8
1.4 Research objective ..... 9
1.5 Outline ..... 11

## 2 Wiring manufacturing drawing design 15

2.1 The EWIS development process ..... 15
2.2 Wiring harness characteristics ..... 19
   2.2.1 The wiring harness digital model ..... 22
   2.2.2 The formboard ..... 23
2.3 Current formboard design process ..... 26
2.4 Shortcomings of current methods ..... 29
2.5 Requirements for an automated process ..... 32
2.6 Discussion: the complexity of formboard design ..... 35

## 3 Development methodology 37

3.1 Design automation technologies ..... 37
   3.1.1 Knowledge Based Engineering ..... 37
   3.1.2 Search ..... 42
   3.1.3 Search and KBE ..... 43
3.2 How to design a design process? ..... 43
3.3 A heuristic approach to development ..... 45
3.4 Application to wiring manufacturing design ..... 49
3.5 Discussion ......................................................... 54

4  Stage-specific multi-state product models 57
4.1 KBE building blocks ........................................... 58
4.2 Requirements for a manufacturing system .................. 63
4.3 Product and process decomposition ......................... 65
4.4 Stage identification ........................................... 69
4.5 Geometry states ............................................... 75
4.6 Product model structure ...................................... 78
4.7 Discussion ..................................................... 82

5 3D analysis and flattening of a wiring harness 87
5.1 Modeling a bundle ............................................. 87
5.2 Wiring bundle deformation ................................. 89
5.3 Modeling methods selection ................................. 93
5.4 3D analysis approach ........................................ 94
5.4.1 3D break-out detection ................................ 94
5.4.2 Bundle flexibility analysis ............................. 99
5.5 Flattening approach ......................................... 106
5.5.1 Existing methods ....................................... 106
5.5.2 Proposed alternative ................................... 107
5.6 KBE implementation ........................................ 113
5.7 Discussion: is this design? ................................. 125

6  Fitting and configuring a flat wiring harness for manufacturing 129
6.1 A multi-objective search problem ........................... 129
6.1.1 Current approach ..................................... 132
6.1.2 Automation challenges ............................... 132
6.2 Parameterization approach .................................. 134
6.2.1 Bend assignment ..................................... 137
6.2.2 Determining bending limits ......................... 139
6.3 Fitting approaches .......................................... 141
6.3.1 Experimentation .................................... 141
6.3.2 Alignment method ................................. 148
6.4 Formboard configuration ................................... 155
6.4.1 Manufacturing ergonomics ......................... 155
6.4.2 Formboard dress-up symbols .................... 157
6.5 KBE implementation ........................................ 159
6.5.1 Bend assignment .................................... 159
6.5.2 Fitting .................................................. 162
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>J Formboard drawing results</td>
<td>281</td>
</tr>
<tr>
<td>J.1 Company validation test harness</td>
<td>281</td>
</tr>
<tr>
<td>J.2 Application performance tests</td>
<td>286</td>
</tr>
<tr>
<td>Samenvatting</td>
<td>293</td>
</tr>
<tr>
<td>Acknowledgements</td>
<td>297</td>
</tr>
<tr>
<td>Curriculum Vitae</td>
<td>299</td>
</tr>
<tr>
<td>Acronym</td>
<td>Definition</td>
</tr>
<tr>
<td>---------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>1D</td>
<td>One Dimensional</td>
</tr>
<tr>
<td>2D</td>
<td>Two Dimensional</td>
</tr>
<tr>
<td>2.5D</td>
<td>Quasi-Three Dimensional</td>
</tr>
<tr>
<td>3D</td>
<td>Three Dimensional</td>
</tr>
<tr>
<td>AI</td>
<td>Artificial Intelligence</td>
</tr>
<tr>
<td>AP</td>
<td>Application Protocol</td>
</tr>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>B-rep</td>
<td>Boundary Representation</td>
</tr>
<tr>
<td>BOM</td>
<td>Bill Of Materials</td>
</tr>
<tr>
<td>BRIC</td>
<td>Brazil, Russia, India and China</td>
</tr>
<tr>
<td>CAA</td>
<td>Computer Aided Analysis</td>
</tr>
<tr>
<td>CAD</td>
<td>Computer Aided Design</td>
</tr>
<tr>
<td>CCW</td>
<td>Counterclockwise</td>
</tr>
<tr>
<td>CG</td>
<td>Computer Graphics</td>
</tr>
<tr>
<td>CLLOS</td>
<td>Common Lisp Object System</td>
</tr>
<tr>
<td>CM</td>
<td>Capability Module</td>
</tr>
<tr>
<td>CSG</td>
<td>Constructive Solid Geometry</td>
</tr>
<tr>
<td>CW</td>
<td>Clockwise</td>
</tr>
<tr>
<td>DEE</td>
<td>Design and Engineering Engine</td>
</tr>
<tr>
<td>Acronym</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
</tr>
<tr>
<td>DMM</td>
<td>Domain Mapping Matrix</td>
</tr>
<tr>
<td>DMU</td>
<td>Digital Mock-Up</td>
</tr>
<tr>
<td>DOF</td>
<td>Degree(s) Of Freedom</td>
</tr>
<tr>
<td>DSM</td>
<td>Design Structure Matrix</td>
</tr>
<tr>
<td>EMI</td>
<td>Electro-Magnetic Interference</td>
</tr>
<tr>
<td>EWIS</td>
<td>Electrical Wiring Interconnection System</td>
</tr>
<tr>
<td>FEM</td>
<td>Finite Element Method</td>
</tr>
<tr>
<td>FBS</td>
<td>Frame Based System</td>
</tr>
<tr>
<td>FPP</td>
<td>Flight Performance and Propulsion</td>
</tr>
<tr>
<td>GDL</td>
<td>General-purpose Declarative Language</td>
</tr>
<tr>
<td>GUI</td>
<td>Graphical User Interface</td>
</tr>
<tr>
<td>HLP</td>
<td>High Level Primitive</td>
</tr>
<tr>
<td>IGES</td>
<td>Initial Graphics Exchange Specification</td>
</tr>
<tr>
<td>INCOSE</td>
<td>International Council on Systems Engineering</td>
</tr>
<tr>
<td>KA</td>
<td>Knowledge Acquisition</td>
</tr>
<tr>
<td>KB</td>
<td>Knowledge Base</td>
</tr>
<tr>
<td>KBE</td>
<td>Knowledge Based Engineering</td>
</tr>
<tr>
<td>KBS</td>
<td>Knowledge Based System</td>
</tr>
<tr>
<td>KE</td>
<td>Knowledge Engineering</td>
</tr>
<tr>
<td>KM</td>
<td>Knowledge Management</td>
</tr>
<tr>
<td>LRU</td>
<td>Line Replaceable Unit</td>
</tr>
<tr>
<td>MDM</td>
<td>Multiple Domain Matrix</td>
</tr>
<tr>
<td>MDO</td>
<td>Multidisciplinary Design Optimization</td>
</tr>
<tr>
<td>MMG</td>
<td>Multi Model Generator</td>
</tr>
<tr>
<td>Acronym</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
</tr>
<tr>
<td>MML</td>
<td>MOKA Modeling Language</td>
</tr>
<tr>
<td>MOKA</td>
<td>Methodology and software tools Oriented to Knowledge based engineering Applications</td>
</tr>
<tr>
<td>MOO</td>
<td>Multi Objective Optimization</td>
</tr>
<tr>
<td>MRA</td>
<td>Main Routing Architecture</td>
</tr>
<tr>
<td>NURBS</td>
<td>Non-Uniform Rational B-Spline</td>
</tr>
<tr>
<td>OEM</td>
<td>Original Equipment Manufacturer</td>
</tr>
<tr>
<td>OO</td>
<td>Object Oriented</td>
</tr>
<tr>
<td>OR</td>
<td>Operations Research</td>
</tr>
<tr>
<td>PDM</td>
<td>Product Data Management</td>
</tr>
<tr>
<td>PIDO</td>
<td>Process Integration and Design Optimization</td>
</tr>
<tr>
<td>PLM</td>
<td>Product Life cycle Management</td>
</tr>
<tr>
<td>RBS</td>
<td>Rule Based System</td>
</tr>
<tr>
<td>RMF</td>
<td>Rotation Minimizing Frame</td>
</tr>
<tr>
<td>SCN</td>
<td>Structuring and Collating Node</td>
</tr>
<tr>
<td>SE</td>
<td>Systems Engineering</td>
</tr>
<tr>
<td>SOO</td>
<td>Single Objective Optimization</td>
</tr>
<tr>
<td>STEP</td>
<td>STandard for the Exhange of Product model data</td>
</tr>
<tr>
<td>SWE</td>
<td>Software Engineering</td>
</tr>
<tr>
<td>TRL</td>
<td>Technology Readiness Level</td>
</tr>
<tr>
<td>UML</td>
<td>Unified Modeling Language</td>
</tr>
<tr>
<td>VEC</td>
<td>Vehicle Electric Container</td>
</tr>
<tr>
<td>VDA</td>
<td>Verband der Automobilindustrie</td>
</tr>
<tr>
<td>VR</td>
<td>Virtual Reality</td>
</tr>
</tbody>
</table>
Knowledge Based Engineering to support manufacturing design

Engineers in Europe have to become more productive if the European aviation industry is to remain leading in the world market. Aviation is an important industry for Europe, which employs large numbers of people and drives high-tech research and development [1]. Competition comes no longer only from North America, but increasingly so from other economies such as the BRIC countries (Brazil, Russia, India, China). Companies like Embraer from Brazil, Sukhoi from Russia and Comac from China have entered the regional airliner market for example. The aviation sector shows the need for more cost-effectiveness and shorter development timescales. Like other technical sectors, the aviation industry will be confronted with decreasing numbers of technically skilled personnel. This is demonstrated by a survey [2] forecasting that in 2016 there will be an imbalance in demand and supply of about 150,000 workers for the technical sectors in the Netherlands alone. As fewer people will have to do more work at lower cost in less time, productivity must be increased.

The challenge of increasing productivity can be greater as the development of aircraft systems becomes increasingly complex. In its vision for 2050 [1], the European Commission identifies complexity management as a key technological domain where improvements are needed in order to secure the competitive advantage of the European aviation industry. Aircraft systems are becoming increasingly complex because of stricter environmental and safety requirements. To achieve optimal aircraft performance developers must consider the entire aircraft life cycle and produce integrated, multidisciplinary designs. The Consideration of manufacturing constraints already during product design is an example. Aircraft development projects follow tight schedules and delays are costly; to mitigate this, timescales for (system) development must be shortened. The need for shorter timescales and the globally distributed supply chain leads to concurrent development of aircraft systems. With many different suppliers, design changes occur frequently and must be managed reliably in order to limit rework and design or manufacturing errors. The
result is that productivity must increase while development complexity increases as well.

Automation and better (re)use of or access to design knowledge are strategies to increase the productivity of the increasingly complex aircraft systems development process. Just as robotics has reduced time spent by workers on repetitive steps in manufacturing halls, automation (i.e. virtual robotics) could similarly limit time spent on repetitive, non-creative engineering work. Complex engineering work cannot always be entirely automated and will require some manual and creative work. In these cases automation can provide support by, for example, presenting the complex problem as a simpler problem, or by providing relevant knowledge for the task at hand.

Generating wiring manufacturing drawings is an engineering task that could benefit from an increase in productivity. In aircraft and aero-engine development, the majority of engineering work (about 90% [3]) takes place during the detailed design and preparation for manufacturing phases. The preparation of manufacturing tooling typically consists of time-consuming and repetitive tasks, with a direct impact on the quality of the manufactured product. This is also the case in the development of aircraft wiring at Fokker Elmo\textsuperscript{1}, the second largest aircraft wiring manufacturer in the world [4]. The following section will illustrate the complexity of wiring development in general while going into more detail on the creation of manufacturing drawings. The latter is the case studied in the F3 'Formboard Fitting by Function' project [5] and presented in this dissertation. Many of these complexities are typical for aircraft system development in general.

1.1 Complexity of aircraft electrical wiring manufacturing design

There is a significant increase in the complexity of electrical wiring in today’s aircraft programs. The past has seen the volume of wiring increase with the introduction of more electrical systems: the Airbus A380 has twice the wiring volume of a Boeing 747-400 (530km [6] vs. 274km [7]). The many different components with their interrelationships and dependency on other systems make aircraft wiring structurally complex\textsuperscript{2}. Because of the use of data-buses for example, the wiring volume no longer increases in recent aircraft programs: Boeing actually reports decreases in

\begin{footnotesize}
\begin{enumerate}
\item Fokker Elmo is a leading company in the design, manufacturing and support of electrical systems and solutions for the aerospace and defense markets. The company aims at growth in sales and profit by reducing cost and lead time, while delivering high quality complex products. In order to remain a market leader, Fokker Elmo aims at increasing productivity through innovation by applying KBE techniques.
\item See the complexity classification by Sussman [8], detailed in section 2.6
\end{enumerate}
\end{footnotesize}
1.1. Complexity of aircraft electrical wiring manufacturing design

total wiring length (6km less wiring in the Boeing 737NG compared to its predecessor [9]). Nevertheless, the level of wiring system complexity is expected to increase further as new electrical systems are introduced and pneumatic and hydraulic systems are replaced by electric systems [10] (i.e. the ‘more electric aircraft’). This leads to a growing number of signals and higher electric power demands. Additional factors contributing to the complexity increase are stricter regulatory requirements [11] on reliability and redundancy on one hand, and the growing customer demand on flexibility of the electrical configuration on the other [6]. Wiring dependency on almost every aircraft system (e.g. for electric connectivity, positioning in the aircraft) is a main reason why wiring manufacturers are confronted with a large number and high rate of changes. The complexity of wiring system development contributed to a 1-year delay of the Airbus A380 program [6] [12].

The Electrical Wiring Interconnection System (EWIS) provides signals and power to every system in the airplane and effectively forms the nerve system of an aircraft (figure 1.1). For installation reasons, it is not possible to have wires running from nose to tail. The system is generally designed and produced as an aggregation of hundreds of wiring harnesses. These harnesses are assemblies themselves of bundled electrical wires and connectors. The EWIS development process starts with the definition of requirements. This is followed by the concurrent definition of the electric architecture (i.e. connectivity and routing schematics) and the modeling process of the wiring system in the aircraft 3D digital mock-up. The outcome of these two design activities consists of a set of electrical and geometrical definitions, which must be transformed into manufacturing instructions and drawings. Then, harnesses are manufactured and finally installed in the aircraft.

Wiring harnesses are produced on flat tables, by means of a 1:1 scale production drawing, typically referred to as a formboard (see figure 1.2). A formboard is a flat representation of the wiring harness Three Dimensional (3D) digital mock-up. The main requirement on the formboard is that a manufactured wiring harness can physically fit in the aircraft as intended in the 3D digital mock-up (see figure 1.3). Not meeting this requirement can cause considerable rework as it results in a wiring harness that does not fit or cannot be installed in the aircraft without using excessive force.

The creation of a formboard is a repetitive and time-consuming process. Figure 1.4 shows the estimated complexity of a formboard drawing versus the time required to create it for a representative set of wiring harnesses. Creating a single drawing can take from a couple of hours to a few days. Note that most wiring harnesses fall in the simple/medium categories, with creation times of around 20
Figure 1.1: Wiring in the Airbus A380 digital mock-up. The A380 electrical wiring system encompasses 98,000 wires, 40,000 connectors, 100,000 signals over 530km of wiring. From [12].

Figure 1.2: A wiring harness on a formboard at Fokker Elmo. Photo by John van Lugtenburg
1.1. Complexity of aircraft electrical wiring manufacturing design

Figure 1.3: The wiring harness is designed in 3D, produced on a flat table and installed in 3D

Figure 1.4: Estimated complexity of formboard drawings versus creation time. [13]

hours. The design is performed using a traditional Computer Aided Design (CAD) package and the working experience of the formboard engineer. Figure 1.5 shows that this process consists of the following steps:

1. Check on quality and completeness of the 3D Digital Mock-Up (DMU) (analysis).
2. Transformation of the 3D model to a flat plane (flattening).
3. Rearrangement of the flat model to fit the given dimensions of a table frame (fitting).
4. Addition of production instructions (dress-up).
These steps bring specific sets of constraints to be respected and involve disciplines such as mechanics and ergonomics. In general, most steps are performed manually, with, as exceptions, the transformation from 3D to 2D and, to a limited extent (at Fokker Elmo), the addition of production instructions. However, while the CAD package automates the 3D to 2D transformation [14] [15], the results are unreliable [16], leading to additional manual checks and adjustments of the model (see section 2.4). The electrical design workbenches available in traditional CAD systems do not sufficiently account for the specific complexities of formboard design, such as the constraints of the physical harness. Because of the large number of wiring harnesses in an aircraft and the many manual quality checks and adjustments needed to create a single drawing, the process is repetitive and time-consuming. Additionally, wiring harness development is subject to many design changes, requiring continuous formboard adjustments or complete redesign.

Considering the time currently required to create a high quality formboard and the increasing volume and complexity of wiring, the productivity of the formboard designers could be improved by eliminating manual, repetitive development steps through automation. Knowledge Based Engineering (KBE) and search methods are proposed as design automation technologies to increase the process productivity.

### 1.2 Knowledge Based Engineering & search

La Rocca defines KBE as: "a technology based on dedicated software tools called KBE systems that can capture and reuse product and process engineering knowledge" [17] p.57. This definition is adopted in this work. The main aim is to reduce development time and costs by automating repetitive, non-creative design tasks and by supporting multi-disciplinary design optimization. The roots of KBE are in the fields of Artificial Intelligence (AI) and CAD [18]. Its cornerstones are rule-based design, object-oriented modelling and parametric CAD [17]. These features of KBE
1.2. Knowledge Based Engineering & search

make it a promising technology to automate the formboard design process.

KBE applications can be developed by constructing a product model using high-level building blocks. In literature such building blocks appear under different names [19] [20], but here the term High Level Primitive (HLP) is used, introduced by La Rocca [17]. As opposed to CAD (low-level) primitives, such as points, lines and solids, a HLP is a functional element or parametric building block, incorporating and reusing relevant knowledge. Figure 1.6 shows HLPs defined for a wing, fuselage or engine. The wing-trunk HLP, for example, incorporates knowledge (design rules) on how to generate the outer shell geometry. The generative product model consisting of these generic HLPs is used to instantiate different configurations of the product [21] [22].

![Figure 1.6: Aircraft HLPs and instances. From [17].](image)

Design is not just generating a product configuration, it is an iterative process during which product improvements are sought. The KBE application itself captures design rules, but is deterministic. Given a certain input a single output will be inferred from on the captured rule set. Design iterations can be performed manually by an end-user changing input parameters, but they can also be automated using optimization or search techniques. Search or optimization algorithms traverse a design space to find a solution that best fits one or more objectives. Multidisciplinary Design Optimization (MDO) provides approaches for the exploration of multidisciplinary design problems.
1.3 KBE supported formboard design challenges

Automation strategies to replace manual work and the unreliable automation methods currently used in the formboard design process must be improved or developed. Existing software packages (e.g. [14] [23] [24] [25]) insufficiently automate design steps and do not account for all relevant manufacturing constraints [26]. Examples of KBE applications to support EWIS electrical design and 3D routing can be found in [27] [28] [29] [30] [31]. However, the manufacturing design process of wiring harnesses has not received so much attention in the scientific community. The process and product knowledge used by formboard engineers to design a quality formboard can be used to develop relevant automation strategies. These strategies must be defined for each process step, as well as for their integration in the overall development process.

The HLP building block concept aims at instantiating a product with a single geometric shape, while a wiring harness can take different shapes during the development process: flat during manufacturing, folded during transport and the desired 3D shape after installation. The different geometric shapes that are imposed on a single product through its development process are called geometric states. The formboard as a wiring manufacturing method requires a Two Dimensional (2D) geometric state, which is constrained by the 3D design geometric state. The building blocks of a formboard design KBE application would need to provide multiple states of a single wiring harness instance. Current KBE implementations using the HLP concept only need to deliver a single geometric state of a product [22] [32] [33]. Other design cases in which a product has different geometric states during manufacturing can be thought of, such as the flat and formed shapes of a sheet metal bracket or flight and jig shapes of an aircraft wing. In such cases, the HLP concept could be extended to model both the design and manufacturing geometric states and manage their interdependencies.

Although KBE has been around for a while [34], the technology has not yet widely been adopted by industry [17]. Exceptions do exist, companies like Rolls Royce and Aker Solutions have dedicated KBE departments. Reasons for the limited impact of KBE in industry are given by La Rocca [34] and Verhagen [35]. An important factor is that the cost and benefits of KBE are often difficult to measure: most KBE publications provide qualitative evidence of the advantages yielded by deploying KBE solutions. However, they generally lack a quantitative assessment including both the gain and in investment required to develop such applications [35]. The early KBE development systems (e.g. ICAD) came with very high price tags and required specialized developers and machines [17]. Companies therefore
tend to invest in other process improvement projects than KBE that come at lower cost and risk. To increase confidence in the capabilities of KBE technology, an effort will be made to quantitatively show the advantages and disadvantages of the KBE implementation.

A KBE research challenge is the limited adherence to a specific development methodology. Verhagen [35] finds that 75% of KBE papers do not use a specific methodology and existing methodologies have limited practical impact. This can be understood from the fact that most KBE activities focus on specific cases, each in a different environment. Where a specific development methodology may be useful in a specific situation, it may prove impractical in other cases. Most existing methodologies regard the development of a KBE application as the implementation phase of a Knowledge Engineering (KE) process [36] [37]. These methodologies typically focus on the acquisition, structuring and modelling of knowledge. Implementation in software is given less attention and often separated from the knowledge modelling and verification steps. This can lead to knowledge errors, misinterpretations, and missing emerging requirements and may ultimately result in an inadequate software application. The field of Software Engineering (SWE) however, focusses on the application development process itself [38]. A combined knowledge engineering and software engineering approach may support effective development of a KBE application while increasing confidence in the final product.

1.4 Research objective

This dissertation objective is to prove or disprove the following proposition:

**Research proposition.** The performance of the formboard design process can be increased in terms of time and quality by largely automating it, using Knowledge Based Engineering.

The validity of the proposition will be assessed by developing a KBE solution to support the formboard process and evaluating the potential of the resulting method. The previous section indicated that automation strategies must be devised for the formboard process, leading to the following research question and sub questions:

**Research question 1.** How to automate the formboard process steps?
1. KNOWLEDGE BASED ENGINEERING TO SUPPORT MANUFACTURING DESIGN

- What expert knowledge governs the formboard generation process?

- How to embed such knowledge in a KBE application that is able to:
  - Analyse a 3D wiring harness for manufacturability
  - Transform a 3D wiring harness model to 2D
  - Fit a 2D wiring harness model within a formboard frame
  - Configure the formboard drawing

- How to integrate the KBE supported formboard process in the EWIS development process?

- What are the results and subsequent benefits of the KBE supported process?

The research to answer these questions was performed in the F3 'Formboard Fitting by Function' research project [5]. This research project by Fokker Elmo and Delft University of Technology, supported by KE-works, aimed at obtaining automatic formboard drawing generation. The project was funded through a Dutch national SRP by AgentschapNL and Fokker Elmo.

Additional to the formboard design study case, two relevant KBE methodological issues have been identified. The manufacturing engineering process imposes specific characteristics on the KBE building blocks, such as the capability to generate different geometric representations of a product. This gives the next research question:

Research question 2. How to model different geometric states of a product and their interdependencies in a KBE application?

And secondly, a methodology is needed to develop the KBE applications to support formboard design. The method should provide confidence that a solution conforms to the unique process requirements of the customer. This provides the last research question:

Research question 3. What methodology can be used to develop a KBE solution that matches customer-specific requirements?
1.5 Outline

Figure 1.7 presents the outline of this dissertation schematically. The top and bottom process flows show the formboard design steps and the preceding and succeeding wiring development process steps in the current and future state (i.e. by deploying the KBE applications developed in this research), respectively.

The current EWIS development process and specifically the formboard process are analyzed and the results are given in chapter 2. Examples of problems that may occur when a formboard is not properly designed are given, as well as requirements for an automated solution.

Chapter 3 describes how knowledge and software engineering development methodologies can be combined into a heuristic KBE development approach for this research. It provides a description of how the approach aims at reducing the complexities of developing a KBE tool. The application to the wiring manufacturing design case is explained.

In chapter 4 the literature on building blocks for KBE implementations are revisited (HLPs specifically) and requirements from the manufacturing design process are discussed. After describing product and process decomposition, an approach for the selection of design stages is presented, followed by how a HLP can feature multiple geometric states. Finally a new product model construction building block is introduced, the Structuring and Collating Node (SCN) class.

Automation strategies for the analysis and flattening of a 3D wiring harness model are proposed in chapter 5. Bundles provide the main degrees of freedom of the wiring harness, which is why their deformation is studied first and a flexibility analysis method is selected. Approaches for the 3D analysis are presented and a new flattening method is proposed. Finally, the automation strategies are implemented in code.

Approaches to fit a flat harness within a formboard frame are given in chapter 6, as well as the main configuration aspects of the manufacturing drawing. The configuration is a multi-objective search problem, and the current methods to solve it are described. The automated solutions require some kind of discretization of the wiring harness model, or bend assignment. An approach to perform fitting is selected and its KBE implementation is described.

Chapter 7 presents the integration of the KBE applications in the EWIS development process. The focus is the main obstacle for integration: the exchange of 3D
1. **Knowledge Based Engineering to Support Manufacturing Design**

![Figure 1.7: Schematic outline of the dissertation](image)

Figure 1.7: Schematic outline of the dissertation
design data. Options are presented and a solution is proposed featuring a custom KBE application that reinterprets geometry imported from a CAD system via the STEP AP214 standard.

Several tests on industrial aircraft wiring harness are performed and the results are given in chapter 8. The performance of the KBE modules is evaluated both in terms of time and quality. A brief note is given as well on the potential future state of the formboard design process, particularly plans to industrialize the KBE modules developed during this research.

Finally, in chapter 9, the research proposition is defended and the research questions are answered. Some recommendations are given for future development.
Chapter Two

Wiring manufacturing drawing design

Automation of a complex process like formboard design requires extensive product and process knowledge. This chapter aims at providing sufficient background knowledge on wiring harness development in order to define requirements for an automated solution. To obtain this knowledge, Fokker Elmo experts were interviewed and literature research was conducted.

Section 2.1 describes the overall development process, after which a more detailed description of wiring harnesses is given in section 2.2. This section also presents the characteristics of the Three Dimensional (3D) and Two Dimensional (2D) representations of the wiring harness: the 3D Digital Mock-Up (DMU) and the formboard drawing, respectively. The design process of the latter is explained in more detail in section 2.3, followed in section 2.4 by an overview of the shortcomings of the methods currently used. With this knowledge requirements for an automated process are set in section 2.5 and the chapter concludes in section 2.6 with a discussion on the complexities of formboard design and the challenges of automating it.

2.1 The EWIS development process

All the wiring harnesses in an aircraft combined comprise the Electrical Wiring Interconnection System (EWIS). This is the infrastructure for electrical signals and power in an aircraft. Its development starts with a set-up phase including the requirements definition, see figure 2.1. The design phase delivers the electrical and 3D design definitions. During the electrical design from-to connectivity specifications are augmented with routing and component details. The 3D design consists of allocating space and routing 3D wiring bundle segments in the DMU of the aircraft. The result of the design phase is complete definition of the electrical system and 3D models of the wiring harnesses. The manufacturing design phase translates the design into instructions and tooling for manufacturing. It is followed by the production of the wiring harness. This mainly consists of routing wires, fin-
ishing endpoints and the application of protective materials. Finally the harness is shipped and installed in the aircraft.

**Figure 2.1:** The EWIS development process, concurrent with the aircraft development process.

**EWIS development within the aircraft development process** The EWIS development process starts concurrently with the aircraft design project preliminary design phase and continues until the wiring harnesses are installed in the airframe. During this process, that can take years, the electrical and geometric designs are developed iteratively. Initially guesses must be made regarding the electrical content and geometric design is limited to space reservations in the early aircraft DMU. As the aircraft design matures and design data becomes available, each iteration produces increasingly detailed electrical and geometric designs. Wiring harnesses are among the first items installed in the airframe, which leaves limited time between the moment when the electrical data is complete and installation. **EWIS development** is therefore a process of continuous change, which actually continues beyond the delivery of the first harness shipment as the **EWIS design** can change per individual airplane.

**Electrical design phase** The electrical design process starts with the collection of connectivity data, also called from-to data, for each electric system. This data is typically stored in system schematics (figure 2.2a). The layout of the main routes for electric signals and power in the airplane is defined in a type of ‘metro plan’. This plan is the Main Routing Architecture (MRA) and the corresponding 2D drawing is commonly referred to as a highway diagram, see figure 2.2b. It is impractical
2.1. The EWIS development process

to run a single wire from one electrical system to another through the entire airplane. To ensure the wiring system is installable, it is split in segments. These splits, called production breaks, are indicated in the MRA and are often positioned at structural boundaries (e.g. wing-fuselage connection). The position of each electric system unit from the system schematics is identified in the MRA. There are different routes a signal can take from one system unit to its destination. These routes are determined for all the signals in the EWIS, taking into account for example length and safety requirements. A connector will have been specified for system units by their manufacturer, but not for production breaks. Therefore one or more connectors must be selected for each production break and signals are assigned to the connector pins. For each electric system, wiring diagrams are created (figure 2.2c), which are effectively the system schematics (figure 2.2a) enriched with harness, route segment and production break descriptions. Several research projects on developing Knowledge Based Engineering (KBE) solutions for electrical design aspects have been performed [28] [27] [29].

3D design phase  As the EWIS system competes for space in the airframe with all other aircraft systems, the 3D design phase must start even though the complete set of system schematics is not yet available. So this phase is performed concurrently with the electrical design. Based on the MRA, space reservations are made in the 3D geometrical environment of the aircraft DMU. Dimensions for the space reservations are typically estimated based on legacy data and the available data from the electrical design. This is followed by a more detailed design of the wiring harnesses in the DMU environment. The work is usually distributed among design engineers by dividing the DMU in zones, enabling them to work in parallel. The design engineer selects and positions harness components and routes wiring bundles through the airframe. Some aspects that are taken into account are: the local environment geometry, separation or clearance requirements, support positions, penetrations of structural elements, bend radius and the need for protective materials. The result of this process is a complete set of wiring harness 3D models, as illustrated for the Airbus A380 fuselage in figure 1.1. Van der Velden [30] developed a KBE application for 3D routing, and his work forms the basis for further research by Zhu [31].

Note that the design process is not always performed by the same company that manufactures the wiring harnesses, but by aircraft Original Equipment Manufacturer (OEM). The same holds for the installation of the wiring harnesses in the airplane.
Figure 2.2: Illustration of a signal A traveling from system box 1 to system box 2. (a) System schematics (b) Simplified MRA of an aircraft forward fuselage. (c) Wiring diagram. Based on [29]
2.2 Wiring harness characteristics

**Manufacturing design phase** The objective of the manufacturing design phase is to enable manufacturing of the wiring harnesses based on the EWIS design. The result from the electrical and geometrical design is translated into a Bill Of Materials (BOM), manufacturing instructions and tooling. As wiring harnesses are manufactured on flat tables, 1:1 scale production drawings (see example in figure 2.1, bottom left), or formboards, have to be created. In some cases the wiring harness cannot be manufactured in 2D; in such cases molds are designed such that the desired 3D shape can be obtained. Some cases can necessitate the construction of a physical mock-up of the aircraft structure, which is very costly. No dedicated research into automating the formboard design process was found in literature.

**Production** Manufacturing a wiring harness consists of assembling a set of standard parts given by the BOM (e.g. wires, connectors) into a complete product. A set of standard parts including marked wires is delivered to the production table where the formboard is prepared for manufacturing by attaching the drawing to the board and inserting pins to hold wires in place. Figure 2.3 shows pins on a formboard. Then, all the individual wires are routed according to the geometry shown on the formboard drawing. This step is called *routing* and is completed when all wires are bound together to form bundles. Often protective materials such as braiding or sleeves are added, which may require the harness to be temporarily taken from the production table to a dedicated braiding room for example. The next step, called *finishing*, is to connect endpoint components to the bundled wires. This is the most time-consuming part of the production process. In this phase, standard parts such as backshells, tubing and shields are assembled, wires are stripped and connected to crimp contacts, which are then installed in the connector body.

This section mentions several typical wiring harness components. Before further detailing the formboard design process (section 2.3), the most important characteristics of a wiring harness will be described.

### 2.2 Wiring harness characteristics

A wiring harness is an assembly of wire bundles and associated components such as connectors. The EWIS system is basically composed of more wiring harnesses that mate to each other or to the electrical systems via connectors.

Figure 2.4 shows a schematic taxonomy of a wiring harness, which includes components such as connectors, backshells, wires, supports, braiding, sleeves, etc. Figure 2.3 shows part of a wiring harness where metal braiding, tape, and angled
2. Wiring Manufacturing Drawing Design

Figure 2.3: Part of a wiring harness on a production table. Photo: courtesy of Fokker Elmo.

Figure 2.4: Wiring harness taxonomy.
backshell and circular connectors can be seen. Figure 2.5 gives an impression of the many different types of components.

A set of wires bound together is a bundle. Wire bundles are bound together by means of tape, tie-wraps or cords for example. A bundle is typically less flexible than its constituent wires. The flexibility depends on the bundle wiring material, the thickness, the binding method and the applied protective materials. In order to increase bending flexibility bundles can be twisted; this reduces the torsional flexibility however. A point where a bundle splits in more bundles is a break-out. Especially for thick bundles, a break-out point can be very rigid.

An endpoint is the assembly of a connector, and sometimes a backshell or adaptors (to attach a shrink boot, shield or conduit tube for example). A backshell can provide shielding from the environment and can be equipped with a strain relief to protect wire contacts for example. Endpoint components exist in different sizes and dimensions as illustrated in figure 2.5; both circular and rectangular components are common. The aircraft environment does not always provides sufficient space to use straight components, which is why angled backshells are often selected, typical angles are 45 and 90 degrees as indicated in figure 2.5. Endpoint components are rigid structures. The relative orientation of a backshell with respect to a connector is set by a clocking angle. A circular connector has a reference point in order to orient it: the keyway.

To protect the harness from chafing, Electro-Magnetic Interference (EMI) and environmental conditions such as moisture and heat, it can be enveloped in pro-
tective materials. Examples of protection materials are braiding (e.g. nomex and metal), sleeves, tubing and tape. Materials such as sleeves and tapes are also used for the identification of the harness or its components. Some of these covering materials affect the flexibility of wiring bundles.

Other components that may be part of or connected to the wiring harness are for instance supports, feed-throughs and structural parts.

Most wiring harnesses have a main branch, from which bundles branch out to connect to system units. This is illustrated in figure 2.6a. The main branch is the thickest and after each break-out point the branching bundles are thinner. Some harnesses have more than one main branch, because of separation requirements for example. As cross-overs are needed between these main branches, so-called closed loops can occur in larger harnesses (figure 2.6c).

![Figure 2.6: Schematic illustration of wiring harness configurations.](image)

### 2.2.1 The wiring harness digital model

The 3D wiring harness digital model consists of the same main components as the real harness. The individual wires are not modeled, but are represented as bundles. Bundles are typically modeled as circular sweeps along a centerline. The centerline curve is in most cases a spline. Bundle centerlines connect to either another bundle or a connector. Connectors, backshells, adapters and other attached components are obtained from component libraries. These components are often solid models with many detailed features, including reference points and curves. Covering material such as braiding, sleeves and tape are modeled by defining their centerline such that it follows a bundle centerline along the section that is covered. A circular sweep with a slightly larger radius than the associated bundle is made along the covering centerline to visualize the covering material.

The main activities of designing wiring harnesses in the aircraft digital 3D environment are: placing connectors, placing support points and routing bundles along support points and around obstacles as explained in section 2.1. Various Computer
2.2. Wiring harness characteristics

Aided Design (CAD) systems are used by industry to perform the digital 3D design. Catia V5 from Dassault Systèmes is popular, but other systems are used as well: Catia V4 and V6, Siemens NX, CADD5, etc. Figure 2.7 shows a digital wiring harness model in the Catia V5 environment. In order to concurrently perform the detailed design of systems in the DMU, the aircraft is divided in zones. Because of this, digital wiring harness models are often segmented as they cross multiple zone boundaries. As the DMU is subject to many changes during the aircraft development process, the wiring digital models change frequently.

2.2.2 The formboard

Manufacturing wiring harnesses on a flat table has proven to be an effective production method. Flat production drawings are used for manufacturing not only in the aviation industry, but also in automotive. The formboard is sometimes called a harnessboard or nailboard. The latter stems from the pins or nails that are used to keep wires in place on the boards (nails can be seen in figure 2.3).

A formboard is a flat 1:1 scale drawing showing the precise wiring harness geometry. It is used for:

- Manufacturing: laying of individual wires from connector position to connector position (routing) and assembly of the endpoint components (finishing).
- Quality control: to check if the completed wiring harness fits the drawing within tolerances.

Both horizontal and inclined (figure 2.8) tables can be used. A table set at a high inclination angle provides an ergonomic advantage during routing [39] and requires less floor space. Finishing and quality control cannot be performed on an inclined table however, horizontal tables are always necessary for these steps. In practice, inclined tables are used only for large wiring harnesses. Depending on the table type a formboard will provide the geometry of all the wiring harness components (horizontal) or the bundles only (inclined).

Figure 2.9 shows the main components of a formboard drawing. The geometric representation of the wiring harness is the most important. The bundles are represented by centerlines and an estimated thickness. The position of components are indicated, sometimes with a tolerance field. The position of a break-out is also indicated by a tolerance field. Endpoints are typically represented by a drawing of the individual components and cutting lines, component identification, and tolerance
Figure 2.7: Screenshot of a 3D wiring harness in the CATIA V5 environment. Image: courtesy of Fokker Elmo.
2.2. Wiring harness characteristics

Figure 2.8: Wiring harness manufacturing at Fokker Elmo on horizontal (left) and inclined tables (right). Photos: courtesy of Fokker Elmo.

Figure 2.9: Simple formboard drawing. Image: courtesy of Fokker Elmo.

fields. Clocking symbols, possibly including a keyway angle show the desired orientation of endpoints components with respect to each other. Unique route numbers allow an operator to identify the path that individual wires must take during routing.

The formboard also shows non-geometrical information for management, quality control and to complete manufacturing instructions. The ruler is used to check the formboard size because dimensions can vary due to plotter tolerance, humidity, temperature. Different information blocks are used to manage an individual drawing; see the identification and change registration blocks in figure 2.9. The legend
block relates symbols on the board to manufacturing standards.

The dimensions of a formboard depend mainly on the size of the wiring harness, floor space and production ergonomics. Smaller drawings have the advantage of limiting the required floor space. The width of the formboard affects the work posture of an operator. For example, a larger board requires an operator to reach further, which can be detrimental to his/her health. Typical formboard widths are 60cm and 120cm as these allow an operator sufficient reach while adopting ergonomically acceptable postures [39]. In case a wiring harness cannot be fully manufactured in 2D, 3D tooling (i.e. mold or physical mock-up) will have to be created. In general the use of 3D tooling negatively affects both production time and ergonomics [39].

2.3 Current formboard design process

The objective of the formboard design process is to translate the digital 3D model into a formboard drawing, while complying to manufacturing and installation requirements.

In the past, before the availability of 3D DMUs, formboard drawings were created by first manufacturing a wiring harness in the 3D physical mock-up of the airplane. The prototype harness was then physically flattened (by force) on a table and a drawing of the contours or a photo were made as a blueprint for series production. As the prototype could fit in the airplane, harnesses built with these drawings would also fit. This last point is still the main requirement for a formboard today. A wiring harness manufactured on the flat board must fit in the airplane.

Figure 2.10 shows the process currently followed to create formboard drawings. The process is performed by a formboard engineer, supported by a manufacturing engineer providing specific manufacturing instructions; it consists of the following steps:

1. Retrieving the wiring harness model from the 3D design team (in-house or external) and checking it for completeness. This initial check includes for example checking whether the components are consistent with the electrical design, all connectors are connected to a bundle centerline, backshells are correctly positioned with respect to a connector, correct sleeve identifications are given, etc.

\[\text{Note that the figure does not account for the creation of 3D tooling. Only the steps resulting in a flat production drawing are shown.}\]
2.3. Current formboard design process

Figure 2.10: The current formboard creation process.
2. Analyzing the 3D model to identify geometries that cannot be flattened. One such geometry occurs when the direction vectors of bundles connected in a break-out point are not all in a single plane. This is illustrated in figure 5.2. Bundle flexibility is analyzed in order to identify shapes that cannot be flattened or are not allowed to be straightened. In the latter case this means that the bundle section can be flattened only if the bend in the 3D model is respected in 2D. When non-flattenable geometry features are found, a change may be requested to the 3D design.

3. (Optional) Converting the CAD model of the wiring harness to the organization preferred CAD format. A design may be provided in a different CAD format because another company can be responsible for the 3D design or the OEM requires the 3D design to be performed using a specific CAD system. Conversions are not always trivial, in some cases this step can require the reconstruction of the entire 3D model.

4. Flattening the 3D model to a single plane. This transformation from 3D to 2D is typically done automatically using a commercial CAD package such as CATIA V5 [14].

5. Checking & adjusting the model to ensure that the flattened model matches the 3D model. The automatic flattening process is not entirely reliable: break-out angles may not be respected for example. More examples are given in section 2.4. As the occurrence of these discrepancies is difficult to predict, the entire wiring harness must be checked for these issues and when necessary adjustments must be made.

6. Fitting the flat model to a frame. Based on the wiring harness dimensions a frame size is selected. The flat wiring harness model is manipulated to fit within the selected frame, without overlaps and using a minimum of bends. This is done by applying bends and adjusting break-out angles. Obtaining a fitting configuration can be quite a puzzle. Complex harnesses require several iterations before a satisfactory result is obtained. If the 3D analysis step gave bends to be respected these must be reproduced as well.

7. Comparing & adjusting the fitted model with respect to the 3D model. All the additional bends and modified break-out angles introduced during fitting must be analyzed for compliance to bundle flexibility constraints. Also, sufficient space must be available in order to add manufacturing information.
8. Adding manufacturing information to the drawing. This includes amongst other the addition of part identification tags, routing numbers, cutting lines, tolerance fields, clocking symbols and more.

9. Checking the completeness and correctness of the drawing for manufacturing.

10. Plotting the drawing.

A survey [13] performed during this research at Fokker Elmo shows that it can take several hours to create a formboard for a simple wiring harness up to weeks for the most complex ones. This is illustrated in figure 1.4. The survey consisted of questionnaires that had to be filled by formboard experts\(^2\) for a set of (only) 15 representative formboard drawings. The time spent on each of the steps in figure 2.10 varied greatly per individual wiring harness. There were some differences between the estimates provided by each expert, which could easily be explained by the experts difference in experience and familiarity to the particular issues of individual aircraft programs. Based on these results it was concluded that none of steps 2 to 9 is, on average, more time consuming than others. Steps 1 and 10 only take limited time.

### 2.4 Shortcomings of current methods

All the formboard process steps, except for flattening and, to a lower degree, the addition of manufacturing information, are performed manually. The manual work consists mostly of operating the CAD system and consulting handbooks/design standards. This process is repetitive and time-consuming because of the hundreds of wiring harnesses in an aircraft and the many manual quality checks and adjustments needed to create a single drawing. Additionally, wiring harness development is subject to many design changes, which impact the process by requiring formboard adjustments or complete redesign. Many of the issues that are checked only occur occasionally, but to guarantee the product quality every single wiring harness must be thoroughly inspected.

#### 3D analysis

The 3D analysis requires manual inspection of all points of interest in the 3D model. An engineer visually identifies geometric features in the wiring harness that may not be flattenable. These features are evaluated with respect to flexibility limits provided by tables or design standards given in handbooks. The

\(^2\)Senior Fokker Elmo formboard designers and manufacturing engineers.
thoroughness of the analysis may vary per individual engineer, and occasionally an issue may be overlooked. The CAD systems currently used do not automatically detect 3D break-outs nor sections with high curvature that can not be straightened or flattened. Actually, some CAD systems do allow the detection of high curvature sections, but based on a simple factor of the bundle diameter: \( R_{\text{min}} = nD_{\text{bundle}} \). In practice, wiring harness manufacturing engineers always use (proprietary) methods, based on years of wiring engineering experience, rather than commercial CAD features.

**Flattening** Although the flattening process is automatic, its results must be checked manually as the flattening result is generally not reliable. CATIA V5 [14] for example, provides an electrical toolbox which includes a method for flattening wiring harness 3D models. However, this flattening method is based on a projection algorithm, hence the orientation of bundles with respect to each other depends on the selected projection plane. The effect of using a certain flattening plane is illustrated in figure 2.11 (only schematic representations are given, as screenshots of issues for actual wiring harnesses can not be published for confidentiality reasons). Suppose that the harness projected in the X-Y plane (right) represents the closest match to the 3D model. If the Y-Z or X-Z planes would have been used instead, the bundle and component orientations would have differed more from the 3D model than strictly necessary to obtain a flat result. The problem is that, depending on the flexibility of the wiring bundles, an orientation too different from the 3D model could lead to installation problems as is illustrated in figures 2.12 and 2.13. These examples show that not accounting for component orientation and flexibility can result in bundles that need to be bent beyond allowable limits, or result in insufficient length to reach a destination connector.

The use of a projection algorithm is not the only cause of the many checks and adjustments that must be performed after the automatic flattening process: high curvature sections identified during the 3D analysis that must be respected are not automatically flattened. In some cases, the flattened model is not exactly positioned in the target 2D plane, which may lead to too short representations of bundles. The correct position and orientation of components connected to the wiring harness model may be lost during the transformation. And the division of wiring harnesses in zones induces additional work as every section must be flattened separately.

**Fitting** The flat wiring harness model is manually fitted within a frame. As bends are introduced and break-out directions are modified, the fitted result must be compared to the 3D model, taking into account the flexibility limits prescribed in
2.4. Shortcomings of current methods

3D model

flattened results (projection method)

Y-Z plane  X-Z plane  X-Y plane

Figure 2.11: Schematic representation of a 3D wiring harness and resulting flat representations, depending on the projection plane.

Figure 2.12: Schematic representation of a 3D wiring harness, the resulting flat representation and the harness during installation. Due to limited bundle twisting flexibility, endpoint A may not be able to reach its mating endpoint.
Figure 2.13: Schematic representation of a 3D wiring harness, the resulting flat representation and the harness during installation. The incorrect orientation of bundle B leads to the situation where there is insufficient length to reach the destination connector.

the design standards or handbooks. All of these steps are not automated in current CAD systems, nor do these systems provide any feedback on the impact of the designed configuration on manufacturing ergonomics. The iterations needed for more complex harnesses can make this a lengthy process.

**Addition of manufacturing information** Limited functionalities to automatically add manufacturing information are available or have been custom made for the CAD system. However most of the information and symbols must still be added manually. As there are layout differences between aircraft programs the automated functions are not always reusable.

### 2.5 Requirements for an automated process

Although with certain shortcomings the current process is able to meet the requirements for manufacturing high quality wiring harnesses. The requirements for the automated process should be the same as for the current manual process. Many of the requirements are stated explicitly in handbooks, more can be deduced from the design steps in manuals and some are implicit in the manual approach to design formboards. Systems Engineering (SE) [40] provides tools useful for the identification of requirements, in particular functional analysis and requirements discovery. Chapter 3 provides more details on this. This section presents the main, high-level requirements, see [41] for more details.

The performance of the formboard design process is measured in terms of cost, drawing design time, and manufactured product quality. Manufacturing time is affected by the drawing layout: for example assembling a connector may take more time when it is hard to reach on the formboard. There is a balance between these
performance indicators: in the current process, spending less time on optimizing a formboard drawing for manufacturing could lead to higher costs due to the need for extra manufacturing resources. Compared to the current situation in industry, the performance requirements for the automated process are:

- At least equal or better product quality,
- Reduction in drawing design time,
- Reduced drawing design cost.

Constraints for the formboard process are provided by the processes preceding and succeeding it. These constraints give rise to functional constraints, as illustrated in the upper and lower part of figure 2.14, respectively.

- Installing the wiring harness in the aircraft poses constraints on the manufacturing method and hence on the formboard drawing. The installation effort and the damage resistance to bending constrain the allowed flexibility of the wiring harness.

- Next to the installation effort, it is obvious that the wiring harness dimensions must be respected for it to fit in the airplane. This leads to the requirements of identifying and respecting wiring harness components geometry and orientation when creating a formboard drawing. A formboard must properly represent the wiring harness geometry and other information (see section 2.2.2), using clear symbols for both manufacturing and quality control.

- The configuration of the formboard drawing affects the required production effort. Operators produce more efficiently when manufacturing ergonomics are taken into account. Straightening bundle segments limits the number of pins that must be inserted in the board and limiting the board dimensions requires less production space. It is easier for a manufacturing operator to assemble wiring harness components if they are easy to reach and when there is ample space available to place manufacturing tools, especially around connectors.

- The formboard process should not depend on a single CAD format, as different OEMs use different CAD systems. A design change can constrain the formboard design as there may be a need to respect an existing formboard design, possibly already used on the production floor.
2. Wiring Manufacturing Drawing Design

Figure 2.14: The process provides constraints which lead to functional requirements for creation of formboard drawings. Adapted from [41]
2.6 Discussion: the complexity of formboard design

The creation of a formboard drawing is complex mainly because a high quality standard needs to be maintained, the result is that it is a repetitive, time-consuming process. In chapter 1, complexity is used as a general term. To clarify what aspects of formboard design actually make it complex, the complexity categorization proposed by Sussman [8] is used. This categorization works well to describes both the complexities of the process to be automated and the process of developing an automated solution itself, as will be discussed in chapter 3.

- **Structural complexity**: exists when a system consists of a large number of interconnected parts. It is reflected in the large number of interconnected, interrelated components in the EWIS and individual wiring harnesses it is comprised of. There are also relations between the different states of a wiring harness, further increasing the structural complexity. The bending limits of the flat model bundles for example depend on the geometry of the 3D model.

- **Behavioural complexity**: exists when the output or behaviour of a system is difficult to predict. The assessment of the mechanical properties of a bundle, which depends on many manufacturing process factors (see section 5.2) is one example. In practice it takes (years of) experience to evaluate whether it is allowed to flatten certain wiring harness geometries (= to predict the behaviour of the flat wiring harness in the 3D aircraft during installation). The less experience, the more complex this is for a formboard designer.

- **Nested complexity**: complexity of the interactions of a system within another system. Formboard design is part of wiring harness development, which itself is part of the aircraft development process. Changes in the aircraft induce changes in the EWIS design and subsequently the formboard design. Anticipating and adapting to these changes from the parent process brings an additional level of structural complexity to the process. An example is the CAD input provided by 3D designers: a new version may be nearly identical to an earlier one, but the CAD model structure can be different. This can be due to design changes but also to CAD format conversions. Assessing the impact of the changes on an existing drawing design (= the interactions between versions) can be so difficult that the entire formboard is redesigned.

- **Evaluative complexity**: the complexity of decision making in an environment where different stakeholders view different aspects of system performance in different ways. Stakeholders in formboard design are the formboard engineer, manufacturing engineer, production floor manager and production
worker. for example. Each of them have a different perspective on the quality of a formboard, size and the features that should be represented on it.
Chapter Three

Development methodology

The previous chapter provided a set of requirements for automating the formboard process. In section 3.1, the two technologies employed to develop the software solution to automate this process are described, namely Knowledge Based Engineering (KBE) and search methods. The complexity of "designing a design process" is explored in section 3.2. Elements from the fields of Systems Engineering (SE), Knowledge Engineering (KE), Software Engineering (SWE) and existing development methodologies are combined into the heuristic development approach (section 3.3) applied to the formboard design case. The application of the approach in practice is described in section 3.4 and subsequently discussed in section 3.5.

3.1 Design automation technologies

3.1.1 Knowledge Based Engineering

Knowledge Based Engineering can be described as an automation technology to support the engineering design process. In a typical engineering design task a product model is generated, transformed and/or analyzed. A KBE application will largely automate these engineering steps based on knowledge about the product and the development process, providing benefits in terms of time and quality. The common denominator of the different definitions of KBE to be found in literature [36] [42] [35] is the capture of engineering knowledge in software systems to assist in engineering design tasks. KBE is best defined as a technology, and here the definition by La Rocca [18] is adopted:

"KBE is a technology based on dedicated software tools called KBE systems that can capture and systematically reuse product and process engineering knowledge."

La Rocca [18] extends this definition with the main objectives of KBE: to reduce time and costs of product development. This is achieved by automating repetitive
and non-creative design tasks; and enabling multidisciplinary design optimization throughout the design process.

KBE is not a new technology, its story starts in the 1970s with the emergence of Knowledge Based System (KBS) or expert systems in the field of Artificial Intelligence (AI). A typical KBS consists of an inference mechanism and a Knowledge Base (KB). When the former is given a problem, it can derive an answer based on the knowledge stored in the KB. This knowledge can be represented by rules and/or frames, leading to two types of KBS: the Rule Based System (RBS) and the Frame Based System (FBS), respectively. Rules are typically in the form of IF-THEN constructs. Frames are concepts with attributes and operations that can relate to other concepts. It can be seen as the application of the Object Oriented (OO) paradigm to KBS. KBS have been successful in many fields such as planning systems and diagnostic tools [18], except engineering design. KBSs are limited to reasoning about facts, which gives rise to the two main reasons for the limited impact of KBS in engineering design [18]: they are unable to perform data processing tasks and do not feature geometric modeling capabilities.

A KBS enriched by the geometry handling and data processing capabilities of Computer Aided Design (CAD) and Computer Aided Analysis (CAA) tools is a KBE system. This is actually how KBE is often described in literature, to quote Chapman and Pinfold [43]: KBE "represents a merging of OO programming, AI and CAD
3.1. Design automation technologies

As illustrated in figure 3.1, similar to KBSs, KBE is supported by the field of KE (appendix D), which again is part of Knowledge Management (KM). Milton [44] classifies KBE as one of the 'Knowledge Technologies', or "computer-based techniques and tools that provide a richer and more intelligent use of information technology".

**Benefits**  The main benefit of KBE is the reduction in time spent on repetitive engineering tasks (i.e. engineering drudgery). Although only a limited number of scientific publications on KBE case studies present quantitative results with respect to saved engineering lead time (see examples in table 3.1), all cases demonstrate the advantage of applying KBE in terms of time. Unfortunately it is not always entirely clear whether the reduction is in lead time, or in man-hours and whether computing time is included or not.

There are more potential benefits of using KBE next to reductions in lead time and man hours. The reduction in time to create a design provides an opportunity to explore more design options, i.e. a larger part of the design space. Stokes [36] indicates the time an engineer can spend on creative work will increase from only 20% to 80% using KBE. KBE provides the resources or frees more time to assess a design on more requirements and different disciplines, leading to more mature and higher quality designs. A designer’s knowledge about a concept product is increased, enabling better design decisions to be made earlier in the overall development process. As the KBE application captures and reuses expert knowledge on how to design a product, a novice can become able to do the same job as his expert colleague. Formalizing expert knowledge into KBE applications enables a company to retain knowledge that might otherwise be lost with an expert leaving. In some cases, a KBE model can serve as a ‘single source of truth’ and ensure consistency across disciplines. The KE process used to develop a KBE application induces communication, knowledge sharing between experts, clarifies procedures and decisions and leads to standardisation. Design outcomes will be consistent and systematically compliant to specified constraints [45]. As such, the use of a KBE application allows consistent generation of high quality designs.

**Applicability**  Now the question is why not using KBE for every engineering design problem. Available literature on KBE [45] [18] [36] provides some advice on engineering design cases or design tasks that are suitable to KBE. The design task should be sufficiently complex to warrant the use of KBE. For straightforward tasks using KBE would be cracking a nut with a sledgehammer, in those cases simpler automation solutions should be applied. KBE should not be applied when a design case:
### Table 3.1: Overview of some KBE projects and the achieved quantitative process improvement

<table>
<thead>
<tr>
<th>Project</th>
<th>Improvement</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aircraft design concepts</td>
<td>from 1 concept generation to 60</td>
<td>[45]</td>
</tr>
<tr>
<td>Windscreen wiper design</td>
<td>from weeks to minutes</td>
<td>[45]</td>
</tr>
<tr>
<td>Home improvement manufacturing</td>
<td>from 4 weeks to 1 day</td>
<td>[45]</td>
</tr>
<tr>
<td>Jaguar bonnet design</td>
<td>from 8 weeks to 20 minutes</td>
<td>[36]</td>
</tr>
<tr>
<td>Wing box redesign</td>
<td>from 8000 hours to 10 hours</td>
<td>[36]</td>
</tr>
<tr>
<td>Airfoil shape optimization</td>
<td>from 2 months to 4 days</td>
<td>[46]</td>
</tr>
<tr>
<td>Compressor design</td>
<td>from 10 days to 1 day</td>
<td>[46]</td>
</tr>
<tr>
<td>Aircraft movable FE model generation</td>
<td>80% time reduction</td>
<td>[47]</td>
</tr>
<tr>
<td>Automotive mesh generation</td>
<td>from 15 weeks to minutes</td>
<td>[43]</td>
</tr>
<tr>
<td>Hot forging process design</td>
<td>from weeks to hours</td>
<td>[48]</td>
</tr>
<tr>
<td>Blended Wing Body analysis</td>
<td>from 1 configuration analysis to 30</td>
<td>[44]</td>
</tr>
<tr>
<td>Fuselage panel design</td>
<td>from 1 solution to 15</td>
<td>[44]</td>
</tr>
<tr>
<td>Aircraft Main Routing Architecture (MRA) signal routing</td>
<td>from 3 weeks to 40 seconds</td>
<td>Fokker Elmo&lt;sup&gt;a&lt;/sup&gt;</td>
</tr>
<tr>
<td>Semi-submersible design</td>
<td>from weeks/months to one day</td>
<td>[50]</td>
</tr>
<tr>
<td>New aero engine fan design</td>
<td>50% time reduction</td>
<td>[51]</td>
</tr>
</tbody>
</table>

<sup>a</sup>“What we used to do in 3 weeks, we do now in 40 seconds” says Theo Tetteroo, former Innovation Manager at Fokker Elmo, on the KBE application for signal routing from their KBE supplier KE-works
3.1. Design automation technologies

- Cannot be abstracted into standard components,
- Is mainly creative,
- Features changing technology,
- Has a chaotic design process.

KBE applications can typically deal with changes as long as these fit within the abstract classes of the KBE model. The organization must be willing to invest sufficient resources in KBE development. The potential benefits are large, but also require significant resources in terms of cost and time of essential -senior- experts. The impact of the latter should not be underestimated as these experts typically fulfill critical roles in a company’s business process. A typical engineering design process suitable for KBE is [18]: "highly rule-driven, multidisciplinary, repetitive and demands geometry manipulation and product (re)configuration".

**KBE applications**  KBE applications are developed by means of a KBE system. A KBE system provides a general purpose engineering language supporting the OO paradigm (but not necessarily restricted to OO) without any domain specific knowledge except on handling geometry. Runtime caching and dependency tracking are also typical features of a KBE language. These and other characteristics of a KBE system (detailed in appendix A) enable generative design.

A KBE application consists of a network of classes, representing a generic product, called the product model. After assigning a set of input values to the product model parameters, a user request for some output can be given, which makes the KBE system evaluate the rules in order to create a design solution. This is a deterministic process: when the same inputs are given to the product model, the same output will be generated, based on the formal and traceable rules of the application. Note that these rules are more than IF-THEN statements, there are: logic rules, math rules, geometry manipulation rules (including parametric modeling rules), configuration selection rules (i.e. topology rules) and communication rules. [18] These applications are typically not stand-alone, but operate as components of larger engineering frameworks that can include, for example, databases, (external) knowledge bases, legacy or commercial analysis tools, Product Data Management (PDM) and Product Life cycle Management (PLM) systems.

There are a number of commercial KBE systems available. AML by Technosoft [52] and GDL by Genworks [53] [54] are both Lisp based KBE languages with built-in CAD capabilities. Knowledge Fusion by Siemens [55] is a KBE system directly
coupled to the Siemens NX CAD system. Such a KBE system is more focused on geometry and has limited data processing capabilities compared to AML and GDL. Other CAD vendors also provide KBE add-ons such as KnowledgeWare by Dassault Systèmes [56], Inventor Automation Professional by Autodesk [57] and Design++ by Bentley [58]. For this work General-purpose Declarative Language (GDL) [53] was selected as the KBE system for development. The main reason for this is that GDL features all the characteristics of a KBE system as described in appendix A.

3.1.2 Search

KBE applications can be used to support product design or transformation processes. As mentioned, a KBE application is deterministic and will provide the same result given the same input parameters. Design however, is an iterative process searching for 'better' solutions. This role can be taken by the KBE application end-user, but can also be automated using search or optimization techniques, which is the topic of this section.

Typical engineering design processes feature coupled and conflicting requirements and require trade-offs between different solutions, which leads to iterations. A design is synthesized by iteratively changing and fine-tuning design variables until an acceptable design is found. Search techniques are mathematical techniques widely used in the fields of AI [59] and Operations Research (OR) [60] that allow the automation of this (possibly informed) trial and error process. The purpose of search is generally to find an (engineering design) solution within a certain design space. This may be an optimal solution, but for many practical engineering problems the important objective is to find a feasible solution and not necessarily the optimal one. This was recognized by Herbert Simon who coined the term 'satisficing' [59] and also by Schut [19], who calls it 'feasilisation'.

Russell and Norvig [59] define search as "the process of looking for a sequence of actions that reaches the goal". Not only the goal state, but also the sequence of actions to get to the goal state are results. How to traverse the design space (or 'look') depends on the search strategy selected. For most engineering problems the goal state is the only relevant outcome and the actions to get there are irrelevant. To this purpose a design problem is formulated mathematically in the form of an optimization problem with an objective function, equality, inequality and side constraints. Various search or optimization techniques can then be applied to minimize the objective function. An introduction to these techniques and Multidisciplinary Design Optimization (MDO) is given in appendix B.
3.1.3 Search and KBE

Both KBE and search methods can be used in combination to automate the design of a complex product. Search methods can be embedded in a KBE application module (e.g. to find the best design option for a part), or can be part of a framework controlling the KBE application, such as the Design and Engineering Engine (DEE) (see appendix B). Some engineering activities do not involve any search activities at all, which is actually the case for some of the design activities in formboard design. As 'design' is often defined as an iterative search process, an engineering problem without any search activities should probably not be classified as 'engineering design', but just plainly as explicit 'engineering'. Having introduced KBE and search as our main automation techniques, the question of how to design a design process can be addressed.

3.2 How to design a design process?

To partially or fully automate an engineering design process is itself a design activity. This chapter explores the design process of a design process which is to be automated through KBE and search methods. The existing design process must be modeled, analyzed and redesigned for improved performance and to fit within the overall development process. This is a complex process, as lack of knowledge and uncertainty make "design processes among the most difficult processes to understand and thus modeling them fraught with ambiguity" [61] p.65. McConnell [38] states that the 'primary technical imperative' of software development is to manage complexity. Sussman's [8] complexity classification for socio-economic systems was used in section 2.6 to describe the complexity of the formboard design process. Here the classification is used to illustrate the complexity of designing an automated solution for such a design process:

- **Structural complexity.** The number and interrelatedness of product and design process components under consideration will be inherited by the various formal and informal models needed for development, leading to modeling complexity.

- **Behavioral complexity.** Adopting KBE means that the automated design process support will be more than a simple emulation of existing process steps. From the start and during development, gaps and inconsistencies in the existing knowledge will be found and new knowledge (i.e. alternative engineering methods) must be acquired or developed. Changes, incomplete knowledge,
ambiguity in understanding and experimentation results cause the emergence of knowledge and requirements during development, leading to behavioral complexity.

- Evaluative complexity. Multiple stakeholders are involved in an engineering design process such as design engineers, manufacturing engineers, domain experts and management. Additional stakeholders that design the KBE supported process are knowledge engineers and software developers. This leads to evaluative complexity, as the performance of the automated process will be evaluated differently by these stakeholders.

- Nested complexity. Largely automating a process leads to it being redesigned. The elimination of manual activities or certain tools from an existing process for instance, means that a designer will have to perform different activities and hence the design process will be structured differently. The redesigned process must fit within an existing business process, which leads to nested complexity.

What methods are available to tackle effectively such a complex problem? Formal methods for the management, analysis and verification of complex systems are needed, which are provided by the field of SE (see appendix C). The field of KE (appendix D), gives methods to elicit and formalize domain-specific knowledge. As the goal is to develop a piece of software, the field of SWE is useful as well providing software development approaches (appendix E). Existing KBE development methods typically consist of the following phases (see figures 3.2a and 3.2b):

A. An analysis phase, in which opportunities for the application of KBE are identified and justified.

B. A development phase, consisting of:
   - Knowledge Acquisition, to capture and validate expert knowledge
   - Knowledge structuring, formalization
   - Knowledge application, i.e. implementation in KBE modules

C. Implementation, use and maintenance of the solution.

A more detailed description of KBE development methods is given in appendix F.
The focus of existing KE/KBE methodologies such as CommonKADS [37] and Methodology and software tools Oriented to Knowledge based engineering Applications (MOKA) [36] is mainly on the Knowledge Acquisition (KA) and structuring phases. Although this is understandable as KA and structuring were considered to be the most difficult phase in KBS design [62], it neglects other important aspects such as the implementation in software. Furthermore, it is not realistic to expect that a specification for an existing engineering design process can be unambiguously turned into a complete KBE application for two main reasons. The first is that acquired knowledge will inevitably be incomplete, i.e. knowledge will emerge at later stages. The second reason is that some engineering steps are flawed and automation may require different approaches to otherwise manual tasks, such as the use of search methods. The engineering process must therefore be redesigned to a certain extent, based on emerging knowledge and new methods.

The potential of using (prototype) KBE application modules to support the acquisition and verification of knowledge is not used by the existing methodologies. There is evaluative complexity in the knowledge verification process. The interpretation of the KB views by experts, knowledge engineers and programmers may be ambiguous. Using KBE modules to verify knowledge has the advantage of confronting an expert with elements more closely related to a final product than knowledge base diagrams and tables, improving the feedback reliability. Existing methodologies assume complete verification of knowledge in (in)formal knowledge bases before it can be implemented in a KBE application. Using MOKA for example, the knowledge implemented in a KBE application is a subset of the formal knowledge base, which is again a subset of the informal knowledge base. As such significant effort is put in validating knowledge that may not find its way into the KBE application. In lean terms this knowledge could be considered to be waste. Entirely removing this waste cannot practically be done as development is an exploration exercise. Adopting a development approach expecting change and allowing uncertainty in the knowledge to be implemented may limit this waste.

The KE process of figure 3.2a is adopted as the overall development approach as explained in section 3.3 and the knowledge acquisition, structuring, application and integration steps (2-5) are further detailed.

3.3 A heuristic approach to development

A heuristic approach to develop KBE solutions is proposed for this research based on a proper combination of SE, KE and SWE methods and principles. The KE process template of figure 3.2a is used as the starting point for the approach presented
3. Development Methodology

Figure 3.2: The KBE development business process (c) and the corresponding elements in two other KE processes (a) [27] (b) [63].

in this section as shown in figure 3.2c. After a general KA, key requirements are selected and KA, implementation and verification activities are performed in an iterative, integrated manner. The objective is to efficiently get to an automated solution matching essential validated requirements.

**Opportunity identification** The development process is initiated by step A of figure 3.2, during which process improvement opportunities are identified using techniques from the field of KM. One approach for opportunity identification that includes the potential application of KBE is the ‘value scan’ [64]. The result of the value scan may point to different process improvement opportunities than a KBE implementation. When the basic understanding of the design process provided by step A leads to the selection of KBE as process improvement, the development can proceed (step B).

**KBE solution development and deployment** Step B is the development of a KBE solution. When a KBE application has been developed and validated, it can be deployed (step C). This includes distribution to users, training users and the use of the KBE solution itself. Usage of a KBE solution in industrial daily practice may and probably will trigger maintenance cycles, as indicated by the feedback loop in figure
3.3. A heuristic approach to development

3.2. The rest of this section will focus on step B.

![Diagram of KBE application development approach](image)

Figure 3.3: KBE application development approach. Expansion of step B in figure 3.2

1. **Initial knowledge acquisition and structuring**  
An initial knowledge acquisition and structuring step (step 1 in figure 3.3) will provide the starting knowledge required to initiate development: basic knowledge on the general business process and detailed knowledge on the concerned design process. The objective of this step is to identify the top-level requirements, use cases, typical inputs and outputs, produce concept models of the process and product under consideration, create a concept application architecture and determine development priorities. This collected knowledge is structured and stored in informal and formal knowledge bases which are verified by domain experts. Unlike other methods, the full knowledge base does not have to be fully validated yet. Only knowledge relevant to decisions to be made directly (key requirements identification, risk assessment) needs to be validated.

**Development iterations**  
After validating the resulting knowledge bases, a key process and requirements set is selected (step 2 in figure 3.3). A development and verification iteration then starts (step 3 in figure 3.3) and after acceptance of the resulting KBE module a new key process and requirements set is selected (step B in figure 3.3). During these iterations the project scope is significantly narrowed initially, and expanded during development.

The iterations continue until all processes and requirements within the project scope have been covered or time has run out. A case study is provided in section 3.4, showing each step in practice. Note that this approach corresponds to iteratively going through steps 3-5 of the MOKA life cycle or steps 2-5 of the KE business process, as indicated in figure 3.2. In practice, step 5 of the KE business process corresponds to the final development iteration.
2. Process and requirements selection  To reduce the complexity of finding a solution for the entire design process, the process and the associated requirements can be decomposed. The development & verification step following the process and requirements selection will target a specific set of processes and requirements. From the process decomposition, the most critical (sub)process is selected, for example process B.1 in figure 3.4. Similarly, from all the known requirements applicable to the selected process, key requirements are selected, as illustrated in figure 3.4. The selection is driven by project priorities. Reasons for a selection can be the implementation effort, availability of experts, involved risk and minimum requirements. The selected process and requirements must be reviewed and accepted by experts before proceeding with development. During the review the time budget for the development step must be set. The software development strategy most suitable to the selected process problem, available time and resources is also selected.

3. Acquisition, development and verification  During step 3 of figure 3.3, knowledge acquisition (step 3.1), structuring (step 3.2) and implementation (step 3.3) are performed concurrently, as illustrated in figure 3.5. Very detailed knowledge is gathered during this KA. Use cases and models from developed knowledge bases are presented to domain experts during the knowledge acquisition for verification and to identify gaps. Specific techniques for KA and structuring have been presented earlier in this chapter. Knowledge structuring will typically start by defining the iteration use case, input and output models. While structuring knowledge, redesigning the (key) process and determining automation (KBE modeling and search) strategies, a knowledge engineer will identify missing knowledge as well. This leads to the need of acquiring new knowledge and adapt the knowledge base. Similarly knowledge gaps will be encountered while developing the software application, leading to reconsiderations of the knowledge structuring and/or modifications to the knowledge base, as indicated by the feedback loops in figure 3.5. The verification levels during development (i.e. knowledge-base, solution, code) are similar to the levels of the SE ‘V’ model.
The resulting KBE module is formally verified (step 3.4) during a domain expert review to assess whether it meets the requirements set for the development iteration. Real-world data can make it possible to not only to verify, but also to validate KBE modules. Acceptance of the results ends the development cycle either concluding the project or by initiating a new cycle.

3.4 Application to wiring manufacturing design

This section shows how the development approach described in section 3.3 has been applied to the project of developing the wiring harness formboard design support application [65].

Opportunity identification The formboard process was identified as a potential opportunity for the application of KBE during the KBE project '3Dtestbench' [66], which aimed at automating the electrical design. This resulted in the F3 project proposal [67]. The potential benefits of applying KBE were discussed in section 1.1. At the start of the F3 project, Kosman [68] performed a so-called value scan, which is an approach to identify process improvement opportunities, to confirm the suitability of applying KBE to the formboard process.
3. Development Methodology

Initiation  The initial development and verification phase (step 1 in figure 3.3) was performed by two researchers and a team of wiring manufacturing experts. By performing amongst others interviews, literature study and observations, general knowledge of the wiring harness development process (figure 3.6) and more detailed knowledge on the formboard design process was obtained. An informal knowledge base was constructed using PCPACK [69] of which some views are given in figure 3.7. Other results of this phase are a requirements document [41], a concept KBE application architecture and an overview of development priorities. The results from this first phase were reviewed with domain experts and management and a key process was selected as detailed next.

Iteration I  The 3D analysis and flattening processes (section 2.3) were selected for the first iteration, as illustrated by figure 3.8a. These process steps were selected because the existing analysis approach is time-consuming and the existing flattening method causes many of the issues discussed in section 2.4. It was also considered more practical to investigate flattening before the subsequent fitting process.

Of all the requirements identified to apply to the 3D analysis and flattening processes, a limited number were selected for this iteration. This is illustrated in figure 3.8b. The main requirements to consider were to respect bundle orientation and to flatten with minimum twist angle. These are both requirements that the current CAD package is unable to meet and are essential for a successful alternative method. Requirements like respecting bending limits and endpoint orientations were not considered in this iteration as, though important, these were expected to be less difficult to solve and as such represented less risk. Development started with identifying the desired behavior of the design process, defining the use case. Very
3.4. Application to wiring manufacturing design

Figure 3.7: Views of the informal knowledge base in PCPACK created during the initial knowledge acquisition phase

Figure 3.8: Development iteration I. a) The selected processes, b) Schematic view of the requirements selection, c) KBE application results
detailed knowledge was obtained by performing further knowledge acquisition ac-
tivities. This included formal expert interviews, but also informal visits and other 
communication. A concept design approach was developed, the required tech-
niques were investigated and domain experts were consulted for verification. In 
some cases, (tacit) judgment tasks needed to be translated to formal activities, e.g. 
the case of detecting 3D break-outs. Class and activity diagrams were made and the 
prototype application was developed with the KBE system GDL [53]. The prototype 
was demonstrated to domain experts, proving the feasibility of the flattening ap-
proach (figure 3.8c). It showed that Fokker Elmo’s customer-specific requirements 
could be captured in a KBE application and provide a better solution than the tra-
ditional CAD package.

The verification meeting (step 3.4) and the process and requirements selection 
(step 2) were performed during the same meeting. The reason for this was the re-
stricted availability of experts. Some verification was therefore already done (infor-
mally) before this meeting. The time budget for each iteration was set by scheduling 
the next demonstration date.

**Iteration II** Critical requirements not taken under consideration during the first 
itration include respecting bending limits and endpoint component constraints. 
To ensure that a wiring harness, manufactured in a flat plane, fits in the airframe 
the bending properties of the bundles must be taken into account (see section 2.3). 
Therefore the same processes are selected with an expanded set of requirements 
(see figure 3.9a,b). With the knowledge acquired from the first iteration and addi-
tional knowledge acquisition steps, automation strategies are defined (knowledge 
structuring) and the prototype application is extended. Figure 3.9c shows a test 
case demonstrating for example that bent sections are flattened when necessary. 
The analysis and flattening functionalities are verified by domain experts.

**Iteration III** The demonstration concluding iteration II showed that the most im-
portant requirements for the analysis and flattening processes could be met. The 
process selected next is fitting (figure 3.10a), in which the flat wiring harness must 
be fitted to a manufacturing table frame. Other requirements to be considered dur-
ing this step are removal of bundle crossing violations and respecting bundle ori-
entations with respect to each other (figure 3.10b). Some requirements that were 
discarded for this step include respecting bending limits and limiting the number of 
bends. Both are important aspects, but not strictly necessary to determine a generic 
fitting method. Unlike the deterministic analysis and flattening processes, fitting is 
a search problem with a limitless solution space (see section 6.1). By limiting de-
sign options, heuristic search methods can be used. Different methods have been
3.4. Application to wiring manufacturing design

Figure 3.9: Development iteration II. a) The selected processes, b) Schematic view of the requirements selection, c) KBE application results

Figure 3.10: Development iteration III. a) The selected processes, b) Schematic view of the requirements selection, c) KBE application results
tested and evaluated, as illustrated for the roll-up algorithm (see section 6.3.1) in figure 3.10c.

**Subsequent iterations**  A number of iterations followed in which more experiments with different search methods were performed and the parameterization of the flat harness was improved for fitting. An iteration took place implementing basic dress-up (manufacturing information) functionalities and several iterations led to the development of a working CAD interface. Parallel to all of these, Graphical User Interface (GUI) s were developed for the different applications.

**Final iteration**  The main objective of the final iteration was to validate the capabilities of the complete suite of KBE applications to create formboard drawings. This required the integration of the different KBE modules and some modifications to address requirements that were ignored until then. For this final iteration the verification step aimed at validating the entire system. The results of this final iteration are presented in chapter 8.

### 3.5 Discussion

A combination of KBE and search methods can be applied to automate engineering design processes. KBE applications are able to capture specific product knowledge and search methods enable finding feasible solutions by iterating between designs. A development approach for the design of KBE applications was proposed and applied to the formboard design case. The approach combines methods from the fields of Systems Engineering, Knowledge Engineering and Software Engineering.

The large amount of product/process elements and their interrelations introduce structural complexity. This complexity is managed by using decomposition techniques, SE techniques (e.g. functional analysis, requirements discovery) and knowledge modeling techniques provided by the MOKA methodology for example. By limiting the initial KA and verification to essential aspects and performing more detailed KA while iteratively developing specific modules, less time is wasted on acquiring knowledge not required for the KBE implementation. Errors in judgment are identified rapidly and unsuitable approaches can be discarded quickly.

The iterations result in early emergence of knowledge and requirements during development which might have arisen at a later stage using a non-iterative approach. This reduces the behavioral complexity of the development process as it allows incorporating emerging knowledge in the KBE application at an earlier mo-
ment. The flexibility of the approach allows keeping options open, limiting the impact of changes.

The short iterations have lead to frequent communication with domain experts, resulting in effective verification of models and prototype applications. This addresses the evaluative complexity of having multiple stakeholders during development. An additional benefit of frequently showing results is that experts are more involved, leading to more interest in and more time being dedicated to the KE project. A drawback of the key process and requirements selection is that domain experts are not well able to judge implementation effort, both overestimating and underestimating the impact of certain requirements.

The nested complexity of automating the formboard process within the overall wiring harness development process has been addressed by taking into account requirements from preceding and succeeding wiring harness development processes.

The development approach should be applied to other industrial KE cases in order to evaluate and improve or 'evolve' the approach. This does not mean that the approach is intended to be followed to the letter. As each problem is unique, the principles and tools discussed here should be tuned to specific KBE development case. KBE practitioners should be encouraged only not to present KBE application case results, but also reflect on the development process, especially if it is unconventional.
In chapter 3 the methodology adopted for developing Knowledge Based Engineering (KBE) applications was presented. The purpose of a KBE application is to automatically generate, transform and/or analyze a digital engineering design artifact, thereby achieving time reduction and quality improvements as was described in section 3.1.1. This chapter aims at providing building blocks for the construction of these applications.

A KBE application can be built from a network of classes, constituting a generic hierarchic product model. Section 4.1 introduces the High Level Primitive (HLP) and Capability Module (CM) building blocks as proposed by La Rocca [17] and similar concepts from literature. Section 4.2 discusses the development of a KBE application from a manufacturing design perspective and highlights differences in building block characteristics compared to the design cases of the previous section. The formalization of product and process knowledge leads to a domain ontology from which candidate HLPs are selected, as explained in section 4.3. Different process steps may lead to dividing the design process into separate stages; section 4.4 provides an approach to identify these stages, resulting in stage-specific HLPs.

The product manufacturing system can require a different representation of that product compared to its detail design representation, as is the case for formboard design (i.e. 3D design and 2D manufacturing). Section 4.5 proposes multi-state HLPs to model products featuring multiple geometric representations. Structuring product models is discussed in section 4.6 which introduces the Structuring and Collating Node (SCN), a class that manages interdependencies between HLPs. Although the functionality and convenience of this KBE modeling approach are demonstrated for the specific case of formboard design, this approach is sufficiently generic and flexible to address other engineering cases. Section 4.7 concludes by providing an overview of the KBE building blocks and their applicability to other cases than formboard design.
4. **Stage-specific Multi-state Product Models**

### 4.1 KBE building blocks

To construct a KBE application building blocks are defined that incorporate the knowledge required to design, transform and/or analyze the product under consideration. La Rocca [17] introduces HLPs to construct product models, these are functional elements or parametric building blocks, incorporating and reusing relevant product and process knowledge. HLPs are captured in classes that can be used to constitute a product model. The product geometry is typically central in the definition of a product model, but non-geometrical aspects are included as well (e.g. material type). The support for analyses (i.e. parameter values or functions of parameter values needed by a discipline-specific tool) is provided by so-called CM classes that extract discipline-specific models from the product model.

A KBE system enables the definition of classes able to handle both morphological and topological geometry operations. In the daily practice of many engineering companies geometric product models are constructed manually using conventional Computer Aided Design (CAD) systems. While this may be unavoidable for innovative design, most engineering work is repetitive. Parameterized models accelerate the creation of product designs with some degree of similarity. In its simplest form a parameterized model allows adjusting the value of a parameter without any relation to other parameters. A more advanced parameterization has parameters depending on function evaluations of other parameters. On top of this ‘parameterization pyramid’ parameters are evaluated through more complex morphological and topological rules and associations to other parameters of model aspects. A KBE system enables constructing parameterizations of this last type.

**HLP - building product models** La Rocca [17] defines HLPs for aircraft conceptual design, to provide a designer with an intuitive and effective aircraft modeling system and support their (multi) disciplinary analysis. The term HLP aims at contrasting with simple or ‘low level’ primitives such as points, splines, etc. According to La Rocca, the Object Oriented (OO) concepts of classification, abstraction and inheritance seem to match how engineers perceive the world. The aircraft conceptual design product model is constructed with HLPs and is called the Multi Model Generator (MMG).

HLPs for aircraft conceptual design can be for example a wing-part, fuselage-part, engine and connection-element, as illustrated in figure 4.1. A wing-part HLP can incorporate knowledge (design rules) on how to generate the outer shell geometry, how to generate internal structure topology and component shapes, etc. The HLP can then be reused to represent product components falling in the HLP
Figure 4.1: La Rocca defines HLPs for conceptual aircraft design and implements these in classes: wing-part, fuselage-part, engine and connection-element. Adapted from [17]
typicality range, for example a wing, horizontal/vertical tail, canard, winglet, or a movable surface such as a rudder.

A HLP class encapsulates declarative and procedural knowledge to define a product component, mainly but not exclusively its geometry. The parameters of a HLP determine its Degree(s) Of Freedom (DOF), i.e. how flexible it is as a building block. The range of the parameters determines the typicality range attainable by the HLP. Design procedures are implemented as rules acting on both geometrical and non-geometrical elements. Reference chains (i.e. associative links) between elements and dependency tracking enable automatic reconfiguration of HLP instantiations. These features allow HLPs to be added, removed and replaced like LEGO blocks in order to generate a large variety of aircraft configurations (hence different topologies) and configuration variants (modifying the same parametric model).

CM - evaluating product models A CM enables the disciplinary analysis of a product model generated by HLP instances. A CM is a class containing procedural knowledge on how to extract an aspect view (i.e. discipline-specific abstraction) from a product model. It cannot be instantiated as a stand-alone object, but must be linked to a HLP; it will augment that HLP with a certain capability. Figure 4.2 illustrates typical aspects of a wing-part HLP, for which the procedural knowledge can be implemented in a CM. CMs can perform directly certain analyses/calculations (e.g. fuel volume evaluation) or write reports that can be used as input by external analysis software (e.g. a mesh for a Finite Element Method (FEM) solver).

The HLP and CM concepts have been used to construct a MMG for aircraft design in the ICAD [17] and later in the GDL KBE system [70] [71] [72] [73]. This MMG forms the core of the Design and Engineering Engine (DEE) framework for aircraft multidisciplinary design introduced in section 3.1.2 and illustrated schematically in figure 4.3. An early version was used for example in a European Blended Wing Body design project [74]. MMG applications based on HLPs have also been developed for other design cases than conceptual aircraft design; for example aircraft component design [75], wind turbine design [32] and plastic injection mold design for the automotive industry [33].

The HLP and CM concepts are adopted and further elaborated on in this research, however similar and derived concepts can be found in literature as ex-
4.1. KBE building blocks

Figure 4.2: Capability Modules generate different aspect models for a wing-part HLP instance. From [17]

KBE building blocks in literature  Ledermann [76] uses the scripting capabilities of CATIA V5 to develop a parametric aircraft model where aircraft components are implemented as dynamic objects with associative links between each other. Based on this work, Amadori [20] [77] proposes 'High Level CAD templates' (HLCt) to create geometric models and support MDO. The HLCt is a parametric building block concept very similar to the HLP. HLCt's are implemented in CATIA V5, using the scripting capabilities of this CAD system. HLCt's are topologically selected from a library by an engineer or a script and their shape can subsequently be changed. Where HLPs can feature topological changes themselves (e.g. change in internal structure), this seems not to be the case of an individual HLCt. In Amadori's work, topological changes are introduced by removing, replacing and adding HLCt's. HLCt's are applied to the design and optimization of industrial robots, transport aircraft and micro air vehicles [77].

Amadori sees code writing as the limiting factor for the applicability of the KBE-
system based HLPs, as engineers would have to "wait for higher fidelity primitives" [77]. How this is different for the script based HLCt's is unclear. The use of a conventional CAD system has the advantage of a powerful Graphical User Interface (GUI) for model adaptations, but its scripting capabilities are limited compared to a KBE system (see section 3.1.1). One advantage of using a conventional CAD system such as CATIA is that it can limit the need for model exchanges between KBE and CAD platforms via neutral formats such as STEP, which comes always at the risk of losing model information. However this advantage is negated when CATIA scripts (i.e. powercopies) cannot be reused, which experience at Fokker has shown can be the case after a release change.

Another approach for aircraft modeling is given by Böhnke [78] who maps UML classes to user defined CATIA features. A rule set is scripted to automatically create CATIA instances out of aircraft design UML classes. Danjou [79] proposes to modularize component parts based on geometric resemblance and implements these as User Defined Features (UDF) using Knowledge Fusion and the Siemens NX CAD
4.2. Requirements for a manufacturing system

Most of these KBE building block concepts are based on the same principles, differences are mainly induced by the CAD/KBE system used. But the generative HLP concept by La Rocca best describes building blocks for a code-centric KBE system such as GDL [53]. The next section discusses the requirements imposed on HLPs by a manufacturing system.

4.2 Requirements for a manufacturing system

The cases discussed in the previous section address product design cases, while the case studied here is about the design of a manufacturing subsystem: the formboard. The design part of the product development process needs to deliver the definition and verification of both the wiring harness and the formboard. Figure 4.3 illustrates a DEE framework that generates a product design definition, where a product model including geometry is generated first, based on initial design parameters, subsequently analyzed for various disciplines and finally improved by means of search iterations. A DEE framework with a KBE application at its core can also be used to define the manufacturing subsystem, in which case a detailed design definition will serve as input. Other aspects in which the current study case differs with respect to the KBE cases discussed in the previous sections are presented next.

Product model structure depending on analysis results  Analysis of design definition aspects may be required in order to develop the product model. The use case is to define the manufactured state of the product (the wiring harness on the formboard) and the in-operation state of the manufacturing subsystem (the formboard). These are based on the design definition of the product (the wiring harness in 3D). Different from the situation shown in figure 4.3, for the formboard design case the product model definition is not the first step, to then proceed with the analysis and verification. The definition of the product model structure is depending on the results from analysis and verification (e.g. 2D geometry depends on analysis of the 3D wiring harness). The KBE application will need to support these interactions between analysis modules and the product model, as is illustrated in the DEE of figure 4.4. An analysis may lead to a specific manufacturing method, which determines how to (de)compose the product model (e.g. decomposition of wiring harness bundles in straightenable and non-straightenable sections). Or a product might have been designed as consisting of distinct entities (e.g. skin, stringers) but may be manufactured as a single part, hence requiring a ‘redefinition’ of the product structure.
More than one geometrical product representation  The product model of figure 4.3 generates a unique geometric master product definition from which aspect models (which can be different geometry abstractions) are derived. In the form-board design case there is more than one dominating representation of the product geometry: the 3D design 'as installed' state, the flat 'as manufactured' state and intermediate states. There is not a hierarchical dependency as is the case for the above mentioned aspect models, but consistency is still required between the different, equally important, geometric states. This issue can be addressed by allowing the same HLP to generate different geometry states. In such a polymorphic HLP consistency between the geometry states could be properly guaranteed.
4.3 Product and process decomposition

(De)composition of design stages  Not only the product, but the design process is decomposed as well, leading to KBE applications that automate all, some or a single design process step. So far, most attention was given to the structural and aspect decompositions of a product to construct a KBE application (i.e. HLPs and CMs). The decomposition of the design process (see section 3.3) gives activities that can be implemented in a single or more KBE applications. The combination of process activities implemented in a KBE application is called a design stage. A trade-off must be made between the advantages and disadvantages of having multiple applications to fulfill the design process. Table 4.1 provides a few considerations for this trade-off. The overall objective is to limit the complete automated design process complexity.

In case more than one stage is selected and multiple KBE applications are developed, these can be placed in a DEE framework sequentially, concurrently or nested within another application. Note that a DEE can be nested in another DEE as well, leading to optimization loops at multiple levels. The DEE for aircraft conceptual design (figure B.1) could be seen to feature a division in design stages, starting with a product definition stage and followed by (concurrent) analysis stages. The manufacturing system design phase follows the product design definition phase as illustrated in figure 4.5. The performance of the manufacturing system design can be used to further mature the product definition in subsequent design iterations.

Extending the KBE building blocks  The points made in this section are addressed by extending the HLP and CM concepts [17], in order to automate the formboard design manufacturing design case. When design activities are split and more than one KBE application is to be developed, design stage-specific product models are defined. These are constructed with stage-specific HLPs. In order to provide different geometric representations in the same product model, multi-state HLPs can be defined. The structure of a product model instance may be depending on analysis results provided by CMs (i.e. CM-dependent product model). The next sections will detail these concepts and present how they are used to create a KBE solution for the formboard design use case.

4.3 Product and process decomposition

The Knowledge Acquisition (KA) (see section 3.3) provides product and design process models such as component taxonomies, process flows, activity diagrams, functional requirements, constraints and use cases. These models are the basis to construct an ontology of the formboard design wiring harness product model. An ontology is an explicit, formal specification of entities, their attributes and the rela-
### Stage-Specific Multi-State Product Models

#### Table 4.1: Considerations when performing the trade-off between combining design process activities in a single KBE application or splitting the process in stages with more dedicated KBE applications.

<table>
<thead>
<tr>
<th>Single stage</th>
<th>Multiple stages</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Process activities combined in one application</em></td>
<td><em>Process activities divided over more applications</em></td>
</tr>
<tr>
<td>When activities use the same product attributes. This limits duplicate attributes and computations.</td>
<td>When the activities share a limited number of attributes.</td>
</tr>
<tr>
<td>When there is a strong dependency between activities.</td>
<td>When the dependencies between activities are limited.</td>
</tr>
<tr>
<td>When the activities involve the same product DOF.</td>
<td>When the activities apply to different product DOF.</td>
</tr>
<tr>
<td>When splitting activities leads to complex interfaces.</td>
<td>When combining activities leads to a more complex product model structure (i.e. requires different product model topology).</td>
</tr>
<tr>
<td>When the activities have a similar level of standardization.</td>
<td>When a specific activity is more likely to change (i.e. rule set is unstable).</td>
</tr>
<tr>
<td>When activities are only useful in a work flow combined (i.e. single use case).</td>
<td>When activities could also be part of a differently structured work flow (i.e. multiple use cases).</td>
</tr>
<tr>
<td>When testing/verifying the behavior of the combined activities is more relevant.</td>
<td>When the behavior of each activity can best be tested/verified separately.</td>
</tr>
<tr>
<td>When activities are controlled by the same agent (e.g. operator, optimizer).</td>
<td>When activities are controlled by different agents (e.g. expert and non-expert).</td>
</tr>
<tr>
<td></td>
<td>When an automation solution already exists for one activity.</td>
</tr>
</tbody>
</table>
Figure 4.5: The result of a product design DEE (figure 4.3) is input for the manufacturing system design DEE (figure 4.4). The performance of the manufacturing system is input for subsequent iterations to increase the product definition maturity.

The wiring harness formboard design case encompasses all activities involved in obtaining a formboard drawing from a 3D wiring harness design definition, where each wiring harness instance differs both in topology and shape. The component...
4. STAGE-SPECIFIC MULTI-STATE PRODUCT MODELS

process analysis, functional requirements, constraints

elements, use cases, test cases

wiring harness component taxonomy

Figure 4.6: Illustration of models developed and obtained during the Knowledge Acquisition. These form the basis to construct a formboard design ontology, see figure 4.7.

taxonomy (figure 4.6) provides the basis for defining structural elements of the ontology, their attributes and relations. More relations are added to the ontology, for example by associating functional requirements to structural elements. Figure 4.7 shows a selection from the ontology which was defined using PCPACK [69]. From the ontology a number of HLPs are extracted:

- **bundle** - Set of wire segments, starting and ending at an endpoint or a breakout (see figure 2.4). A bundle has a uniform wire content and its diameter is therefore constant. The bundle is a flexible element that provides the DOF to flatten and fit the wiring harness geometry to a formboard. The most important attributes are the centerline geometry, diameter and flexibility parameters.
4.4 Stage identification

- **endpoint** - Assembly of connector, backshell and adapter components (see section 2.2). The individual components are not selected as individual HLPs because they form a fixed, rigid assembly without any DOF with respect to flattening and fitting. Endpoints can have diverse geometries (e.g. rectangular, circular, angled), but have attributes in common such as orientation planes and bundle connectivity.

- **connection-point** - Virtual point that represents endpoint connections and break-out points, as illustrated in figure 4.7. The introduction of this non-physical primitive reduces the complexity of the bundle HLP, which only needs to interface with a connection-point and not with an endpoint and bundles. It provides the means to analyze break-out points and position bundle and endpoint HLPs with respect to each other.

- **covering** - All components that cover or envelop one or more bundle segments, while following the bundle geometry definition. This includes braiding, sleeves, and also twist. Multiple coverings can be assigned to one bundle segment and these can affect bundle flexibility. Covering was not defined as part of a bundle because it can extend over multiple bundles, require specific representations on the formboard, requires specific manufacturing steps other than routing, and is typically designed as a separate geometric component during the 3D design.

- **attached-component** - All components that are not directly part of the wiring harness, but must be assembled with it. These are rigid components such as supports and structural elements, and may be connected to bundle or endpoint HLPs.

- **formboard-frame** - The frame of the formboard drawing, including features not directly corresponding to any of the wiring harness HLPs (e.g. legend). The length and width are the main DOF of this typically (but not necessarily) rectangular primitive. The geometry is solely represented in the 2D domain.

4.4 Stage identification

The product and process models are mapped to one another to support the trade-off and selection of design stages. In Systems Engineering (SE) relations between entities or functions are often mapped using square matrices, such as the N2 diagram [81] and the Design Structure Matrix (DSM) [82]. A Domain Mapping Matrix (DMM) [83] is rectangular and maps different domains onto each other (e.g.
Figure 4.7: Structural views of the formboard design ontology and the HLPs selected from it.
4.4. Stage identification

functions and product architecture). The DSMs and DMMs can be combined into a Multiple Domain Matrix (MDM) \[84\], which is a square matrix that allows representing relations both within and between domains. Such a MDM for design process activities and HLPs is illustrated in figure 4.8.

![Figure 4.8: Illustration of a multi-domain matrix for activities and HLPs.](image)

The relations between the activities match the activity diagram links (triangles in figure 4.8) and the dependencies between HLPs correspond to those in the product structure (circles in figure 4.8). The top-right matrix shows attributes that are set/computed in a certain activity. The bottom-left matrix indicates what attributes are required to perform a certain activity (i.e. DOF, reference geometry, model visualization, etc.).

![Figure 4.9: Possible stages in the MDM of figure 4.8.](image)
The diagram can be used to identify potential stages in the design process, for which a KBE application can be developed. It can also be used to visualize the relations between design activities and HLPs for an existing KBE application or one under development. In the left part of figure 4.9, the design process is implemented as one stage, in a single application. The HLPs of the KBE application feature all the attributes and procedures associated with the activities. The right side of figure 4.9, illustrates a case where the design process is split in two stages. When using the MDM to analyze and select stages, these are identified by clustering and sequencing activities and subsequently making a trade-off between the complexity of a holistic or a modular solution. For this trade-off the MDM is analyzed while considering the aspects listed in table 4.1. In case the multi-stage scenario is selected, two sets of HLPs are developed according to the product model ontology (i.e. HLP 1 of stage 1 will ontologically match HLP 1 of stage 2). This way a ‘family’ of KBE applications is built, with stage-specific HLPs defined along the same domain-specific vocabulary.

Figure 4.10 presents the MDM for the formboard design process activities and HLPs, corresponding to the final demonstrator applications. The activity matrix consists of mostly sequential steps without feedback, except for the 3D analysis that follows bend assignment, and fitting adjustments after dress-up. The dependencies between the HLPs (bottom-right matrix in figure 4.10) indicate that one or more attributes from one HLP are required to determine the value of one or more attributes of another HLP. Note that these dependencies mainly concern relative geometric positioning in 3D and 2D. In the top-right matrix, attribute values computed during an activity are given; the 3D analysis activity for example determines a set of straightenable/flattenable sections in a bundle HLP. To perform the 3D analysis activity, the bundle HLP must provide amongst others a 3D centerline, as shown in the bottom-left matrix of figure 4.10. Besides reference geometry, this matrix indicates for example attributes representing the DOF of the HLPs as required per activity and different visualization modes of the HLPs. Instead of listing all attributes, a selection is shown and attributes are grouped together for readability (e.g. ‘dress-up visualization’ includes a set of geometric attributes).

The three stages of the formboard design process as implemented in the final demonstrator applications are indicated in figure 4.10 as well. Note that in practice decisions to combine activities in one stage or not, were made per KBE development iteration (section 3.4).

- **Stage 1: CAD exchange.** The ‘import wiring harness definition’ activity obtains the 3D wiring harness geometry from the preceding 3D design process and maps that to the formboard design HLPs, see figure 4.10. The endpoint
HLP for example must incorporate rules to identify and parameterize component types. The input variability is large because different customers use different CAD systems and adhere to different product model structure. The limited standardization means that the application is expected to have a high change rate, probably for each aircraft program. This stage provides the succeeding stage with 3D geometry data in accordance with the formboard design ontology. The approach taken uses the STEP format and reparameterizes the wiring harness model to match the formboard design ontology. It is implemented in the KBE application $f3input$ (section 7.3).

• **Stage 2: Analysis and flattening.** The 3D analysis and flattening activities are combined in one stage because the results of the former provide values for attributes required by the latter. For example, the analysis divides a bundle HLP in sections, which are used to generate a flattened model. Sharing these attributes in a single HLP definition is much simpler than defining an interface between the two. Another factor is that analysis results can be verified with both 3D and 2D visualizations of the analyses. This stage is separated from stage 3 because it could be reused as a stand-alone analysis module in the 3D design definition phase (indicated in the top-left corner of figure 4.10). To eliminate a feedback loop between the fitting activity and the 3D analysis activity, the 'bend-assignment’ activity was introduced in stage 2 (see figure 4.10, in italic). This activity, discussed in section 6.2.1, pre-analyzes the 3D model with respect to potential bends. Stage 2 is implemented in the KBE application $f3flat$ (chapter 5).

• **Stage 3: Fitting and dress-up.** The third stage combines fitting and dress-up activities because these are coupled: outlines of dress-up features are needed to perform fitting and certain dress-up feature parameters depend on the wiring harness layout. The export activity is relatively simple once a dressed-up model visualization is available, including it in this stage is the simplest solution. Additional reasons for separation from stage 2 are that different fitting algorithms, requiring different product structures, can be developed. Both fitting and dress-up are activities taking place solely in the 2D domain, 3D attributes are therefore not relevant and the model DOF (bundle bending, flipping) are different from the stage 2 model. This stage is implemented in the KBE application $f3fit$ (chapter 6).

The attributes relevant for each stage-specific HLP are indicated in the diagram of figure 4.10. The diagram also indicates where interfaces are required and what attributes these must be able to transfer. Note that only a selection of attributes is
4. SPECIFIC MULTI-STATE PRODUCT MODELS

Stage 1: CAD exchange
(f3input application)

Stage 2: Analysis & flattening
(f3flat application)

Stage 3: Fitting & dress-up
(f3fit application)

4.1: MDM of the formboard design process activities and HLPs, and the division in stages as implemented in KBE applications.
4.5 Geometry states

A geometric state is a geometric configuration corresponding to a specific design domain of one product design. Geometry states can represent a physical product shape at different moments in its life cycle. For a wiring harness these are for example the 3D installation shape, the 2D or 2.5D manufacturing shapes and the shipping shape. Intermediate shapes such as the 2D wiring harness before or during fitting also represent distinct geometric states. The distinction between different design geometries and geometric states of a design is illustrated in figure 4.11.

Figure 4.11: Illustration of the difference between a design and its state(s) for a folded box.

There are two approaches to model multiple geometric states in a stage-specific HLP:

1. Parameterizing the geometry in the HLP such that different states can be represented by varying parameters.

2. Including distinct geometric objects in the HLP to represent different states.

Both are illustrated in figure 4.12 for a tower of cubes with a 3D design definition and a 2D manufacturing system requiring. In option 1 the cube HLP geometry is parameterized such that varying geometric shapes can be obtained by changing the 'rotation-angle' parameter. The 3D design and 2D manufacturing states are obtained by setting this angle to 0 and 90 degrees. A single set of geometric children represents both (and intermediate) states: 'face'. Setting the rotation-angle parameter does not change the design, it transforms the design within the DOF of the
Figure 4.12: Illustration of two approaches to model a primitive with multiple geometric states.

Figure 4.13: Simple illustration of the difference between geometry states, aspect models and visualizations.
4.5. Geometry states

designed product (i.e. it is a product DOF, not a design DOF). Such a parameterization may not always be practical: intermediate states are not necessarily relevant, rules defining intermediate states may not be available or require computations unnecessary for the design case, increasing the complexity of the HLP.

Option 2 in figure 4.12 shows a cube HLP where the two relevant states are both included as distinct sets of geometric children: '3D-face' and 'flat-face'. Both states are available in the HLP, without having to change a parameter representing a product DOF. Relations, dependencies between the two states are managed in the HLP, which ensures a consistent design where common attributes (e.g. dimensions) are shared. In case only a single representation is to be displayed, the geometric elements of one state can be hidden by means of a switch (in GDL classes have a :hidden? attribute that can be set to t or nil). Rules with respect to positioning the HLP geometries can be included in the HLP definition, but may also be specified in the parent class. In order to properly position HLPs with respect to each other, the SCN class provides two coordinate systems for relative positioning and orienting, one for the 3D geometry and one for the 2D geometry.

For both options in figure 4.12 the different states are defined and represented in the HLP, independently of a parent class. The HLPs could be implemented in different applications, that can have different product model structures. The HLP contains the rules/knowledge to construct each geometric state. Note that the states can be interdependent, the rules to construct one state geometry can be a function of computations or analyses taking the other state as input. An additional state can be implemented in a HLP directly, but also with a mixin CM.

Different states or representations of a product geometry should not be confused with the visualization or aspect models of the geometry. The difference is illustrated in figure 4.13 for the folded cube example. An aspect model is a domain-specific abstraction of a certain geometric state. The same type of aspect model could be created from different states. Visualization is the manner in which a geometry is graphically displayed. Visualization modes can be relevant for just one or more states and states can be visualized in different manners. Typical visualizations are for example different levels of geometry details. Visualization modes can be implemented by means of settings or rules that hide/show geometric features or modifying display settings. Additional visualization modes may also be added by CMs. Note that a visualization mode can correspond to a specific aspect model. The Methodology and software tools Oriented to Knowledge based engineering Applications (MOKA) methodology does not account for multiple geometric states in a product model, but does account for different visualizations of a product [36] p.214.
4. STAGE-SPECIFIC MULTI-STATE PRODUCT MODELS

The states, aspect models and visualization modes of each formboard design process stage are indicated in table 4.2. The \textit{f3input} application (stage 1) features a single geometry state: the 3D design definition. The \textit{f3flat} application (stage 2) features multiple states implemented according to option 2 in figure 4.12. Option 1 is used in the \textit{f3fit} application (stage 3), where adjusting bundle DOFs allows the wiring harness to be fitted.

Table 4.2: Overview of the states, aspect models and visualization modes of the bundle, endpoint and connection-point HLPs for each stage.

<table>
<thead>
<tr>
<th>High Level Primitive</th>
<th>bundle</th>
<th>endpoint</th>
<th>Connection-point</th>
<th>Illustrations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stage (application)</td>
<td>1. Import (f3input)</td>
<td>3D</td>
<td>3D</td>
<td>-</td>
</tr>
<tr>
<td>States</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Aspect models</td>
<td>Shape abstraction</td>
<td>-</td>
<td>x</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Trimmed centerline</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Visualization modes</td>
<td>input (from CAD)</td>
<td>x</td>
<td>x</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>output (f3 format)</td>
<td>x</td>
<td>x</td>
<td>fig. 8.1</td>
</tr>
<tr>
<td>Stage (application)</td>
<td>2. Analysis &amp; flattening (f3flat)</td>
<td>3D</td>
<td>2.5D</td>
<td>2D</td>
</tr>
<tr>
<td>States</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Aspect models</td>
<td>flexibility analysis model</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td>bend-assignment</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td>twist analysis results</td>
<td>x</td>
<td>x</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>twist analysis results</td>
<td>x</td>
<td>x</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>break-out analysis results</td>
<td>-</td>
<td>-</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td>bends</td>
<td>-</td>
<td>-</td>
<td>fig. 5.25</td>
</tr>
<tr>
<td>Visualization modes</td>
<td>input</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td>output (f3 format)</td>
<td>x</td>
<td>x</td>
<td>fig. 5.20</td>
</tr>
<tr>
<td>Stage (application)</td>
<td>3. Fitting &amp; dress-up (f3fit)</td>
<td>2D (variable)</td>
<td>2D</td>
<td>2D (variable)</td>
</tr>
<tr>
<td>States</td>
<td>2 + intermediates</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Visualization modes</td>
<td>fitting</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td>dress-up</td>
<td>x</td>
<td>x</td>
<td>fig. 6.3</td>
</tr>
</tbody>
</table>

4.6 Product model structure

Stage-specific HLPs are the building blocks with which a KBE application product model can be constructed, but do not define the product model structure they are a part of. Products such as aircraft and wiring harnesses are considered structurally complex because of the many components and their (inter)dependencies. While CAD and KBE applications feature a product tree, the connections between product components actually represent a graph or lattice. This lattice structure can be described by a structural ontology such as given in figure 4.7. For a specific design stage, the ontology is mapped to a tree structure (i.e. an instance of the ontology is made). This mapping is not straightforward as different design phases use different hierarchies of a product, [85] warns that "problems arise from trying to map a complex lattice structure to a tree".
4.6. Product model structure

Depending on which product aspects and relations are relevant during a design stage, different hierarchical aggregations can be used in a product tree. This is illustrated in figure 4.14 for a schematic wiring harness. The top-left figure shows a product model structure based on signal groups, which is useful for routing wires during manufacturing for example. During the 3D design the wiring harness is often divided in zones with main routes (top-right). Zones are not relevant during formboard design, so, for the latter, one of the product model structures at the bottom of figure 4.14 could be used. Figure 4.15 shows the class diagrams corresponding to the product structures of figure 4.14. Note that the class diagrams (tree structures) are each different mappings of the structural ontology of figure 4.7 (lattice structure) on a tree.

To construct a product model, classes are required that aggregate HLPs and CMs and specify their interdependencies. Such classes are not building blocks representing a generic component like HLPs but allow representing the latter in a specific product structure. These classes are called Structuring and Collating Nodes (SCN)\(^1\). Structuring because they allow constructing different product hierarchies using the same set of HLPs and CMs. Collating because they provide links (i.e. the 'glue') between HLPs and between HLPs and CMs. Node indicates that they are represented as a separate entity in a product tree. Multiple levels of SCN classes can be used, i.e. they can be composed of other SCNs. Two specific SCN cases can be distinguished: when a SCN represents the root of a complete product (product SCN); and when the SCN represents the root of the KBE application (application SCN). The product SCN can also be an application SCN.

Both HLPs and SCN classes incorporate topology rules that lead to a certain product tree when instantiated. These rules can be a function of analysis results that are generated by internal or external analysis modules. Internal analyses can be implemented in aspect-analysis CMs; external analyses require report-writer CMs to interface with an external tool. Figure 4.16 illustrates the different ways CMs can be linked to a HLP class. Note that CMs can be linked to SCN classes in the same ways. The dependency of the product model structure on output provided by a CM is indicated by a dependency link between the CM and a composition link, as illustrated in figure 4.17.

In each identified formboard design stage, the corresponding application uses specific product structures that are depending on CMs:

- **Stage 1: CAD exchange.** The wiring harness 3D design definition phase is

\(^1\)The term for this type of KBE class was determined during discussions with Durk Steenhuizen.
Figure 4.14: Schematic illustration of different ways to structure a wiring harness model. The corresponding class diagrams are given in figure 4.15.
4.6. Product model structure

Figure 4.15: Class diagrams for the product models structures of figure 4.14, each respecting the ontology of 4.7.

Figure 4.16: Overview of the different manners in which a CM can be linked to a HLP.
often performed by different companies, using different CAD systems. This leads to a large variation in the structure of the provided product trees, which are often zone-based (see top-right of figure 4.14). The \textit{f3input} application transforms these to useful HLP parameters. As shown in the left part of figure 4.18, each HLP collection SCN is dependent on the STEP-import CM. The CM output drives rules in the SCN classes that determine the HLP multiplicities (e.g. number of bundles), class types (e.g. connector type) and various attributes. More details are provided in section 7.3.

- **Stage 2: Analysis and flattening.** The division of a bundle HLP in bundle-sections is dependent on a CM for bending analysis, as indicated in the center of figure 4.18. The multiplicity, specialization and geometric parameterization of bundle-section objects depends on the results of this aspect-analysis CM. More details are given in chapter 5.

- **Stage 3: Fitting and dress-up.** Both flat and branching product models have been used when experimenting with different fitting strategies. The branching product model structure is illustrated in the right part of figure 4.18. At each branch level, HLPs that are part of a branch are identified and instantiated based on topological rules. Dependency links are shown between the HLP composition links and SCN computer-slots. More details can be found in chapter 6.

### 4.7 Discussion

KBE applications for product design can be constructed by means of flexible building blocks, called HLPs, CMs and SCNs. HLPs and similar concepts in literature cap-
4.7. Discussion

Figure 4.18: Simplified class diagrams of the formboard design KBE applications, indicating certain composition links are dependent on SCN rules or CMs. Detailed diagrams are given in figures 7.4, 5.24, H.2, 6.32 and 6.33.

ture design rules, automate the generation of a product geometry, perform analyses and search for more optimal configurations. To summarize the previous sections a short description of the three KBE application building blocks is given:

**High Level Primitive** A HLP is a reusable KBE class that incorporates declarative and procedural knowledge to generate an engineering product component, including geometry. This knowledge is specific to a HLP and is not directly shared or reused by other primitives [17]. The knowledge associated with a design process can be divided in stages, for each stage stage-specific HLPs can be defined, leading to a family of HLPs. A HLP component features one or more geometry representations (states) and one or more visualization modes. It is not dependent on a specific tree structure, i.e. it can be reused in a different structural context. HLPs can be composed of other HLPs from a subdomain; what one one calls a HLP is in a way quite arbitrary and subjective. For example in a KBE application for conceptual aircraft design a fuselage HLP may feature a seat as a component, while a KBE application for seat design may feature cushion, leg or table HLPs for instance. The definition of HLPs is related to the level of modeling granularity of a given KBE application.

**Capability Module** A CM is a KBE class containing procedural knowledge on how to evaluate or extract an aspect view from a product model. It is not a stand-alone object, it does not represent a component in itself and cannot be instantiated (gen-
generate geometry, answer messages [17]) unless it is linked to a HLP or a SCN class. When linked, it provides methods to one or more HLPs to allow evaluation and/or extraction of a design instance. CMs depend on the product tree structure inasmuch that they are often designed to link with a HLP or SCN in a specific way (e.g. as a child-object).

**Structuring and Collating Node**  The SCN KBE class aggregates HLPs and contains the procedural and declarative knowledge defining their interdependencies. It can also be composed of other SCN classes and CMs linked to HLPs. SCN classes incorporate rules defining the product model structure, which may depend on CM evaluations. They are typically product model specific and are not as reusable as HLPs are. Indeed this is the most volatile element of a KBE application. While defined HLPs and CMs can be shared within totally separate KBE applications, the SCN provides the specific infrastructure of the given application.

**Complexity**  A typical KBE application aims at reducing the complexity of a design process from the perspective of the design engineer. At the same time the complexity of the application and its development (see section 3.3) should be limited. This concerns for example the structural complexity of the application itself: finding the right balance between the structural complexity of few integrated modules versus the complexity of interfacing between many modules. The division in stages also impact the evaluative complexity, such as verifying or validating application behaviors. Adopting multiple stages may limit the impact of changes on the complete application suite. In the formboard design case for example the CAD-exchange application can be expected to change more often than the analysis and flattening application. One way to link multiple KBE application is by aggregating them via one SCN class, effectively producing a single KBE application. Other methods for linking multiple KBE applications are not within the direct scope of this work, some possibilities are an agent-based framework as proposed by Berends [86], using High-Level Activities to model simulation work flows as proposed by Chan [87], work flow management tools, Product Data Management (PDM) or Product Life cycle Management (PLM) systems, Process Integration and Design Optimization (PIDO) tools (e.g. Optimus [88]).

**Other use cases**  The discussion on HLPs, CMs and SCNs has focused on the formboard design use case. The SCN class also applies to other cases such as aircraft conceptual design. See the MMG of figure 4.1 where the GenericAircraftProduct-Model, LiftingSurface, KCA and BWB classes are effectively SCN classes. HLPs with multiple geometric states can also be relevant in other cases than formboard de-
4.7. Discussion

Design. An illustrative manufacturing design case is the development of sheet metal parts where a product is designed in 3D but needs to be manufactured from a single sheet of metal, see figure 4.19. This involves analyses to evaluate the manufacturability, a search for the most optimal sheet topology as detailed by Tai [89] and the creation of manufacturing instructions. Multiple geometric states can also be relevant for an aircraft (conceptual) design MMG. A single wing-element HLP instance could provide different relevant geometry states to perform analyses on, for example: the geometries of both the loaded and unloaded wing (i.e. jig shape) [90]; the different deployed states of high-lift devices or a landing gear. The next chapters will detail the development of KBE solutions for each of the formboard design stages.

Figure 4.19: Sheet metal corner buffer with 20 sample layouts. From [89]
Chapter Five

3D analysis and flattening of a wiring harness

Equipped with the KBE techniques described in chapters 3 and 4, the automation of formboard process steps can be addressed. This chapter presents strategies and an implementation for automating the 3D analysis and flattening steps of the formboard design process (i.e. stage 2 in figure 4.10). During the 3D analysis and flattening steps a digital model of the 3D harness is provided as input, it is analyzed for flexibility issues and transformed to 2D as required for the generation of formboard drawings. The capabilities of current analysis and flattening tools are limited, requiring time-consuming, repetitive manual work. The objective here is to develop and implement automation strategies for analyzing and flattening a 3D wiring harness model.

The methods currently used to create formboard drawings and their drawbacks were treated in sections 2.3 and 2.4. As the flexibility of the wiring harness is the main concern when going from its Two Dimensional (2D) production state to the Three Dimensional (3D) installation state, section 5.2 explores the mechanics of wiring bundle deformation. These sections provide the necessary background to develop automation strategies for the 3D analysis of a digital wiring harness model in section 5.4 and its subsequent flattening in section 5.5. Section 5.6 describes the manner in which these strategies are implemented in a KBE application. The chapter concludes with a discussion of the KBE application capabilities (section 5.7).

5.1 Modeling a bundle

The behavior of wiring harness bundles or similar product like mechanical cables have been modeled or simulated in academia and industry. This section briefly discusses how bundles have been modeled and how the material properties are found.

Methods to simulate cables have been implemented for: animation purposes
in Computer Graphics (CG) and Virtual Reality (VR) applications, where the deformations of a cable must look realistic. In industrial applications like for example a simulation in an automobile Digital Mock-Up (DMU), the mere appearance of realism is insufficient, the cable deformations must represent actual physics in order to evaluate the feasibility of a design. Different approaches to model cables have been implemented. Many approaches (but not all) use Cosserat rods or Kirchhoff rods. The latter are Cosserat rods where the assumption is made that shearing can be neglected. Often stretching of the rod is neglected as well. Note that the mechanics of wire rope [91] describes the behavior of twisted cables, where wires are arranged helically.

A popular method to model cables is with a mass-spring system [92] [93] [94] [95]. Grégoire [94] for example uses this method and solves the simulation with an energy minimization algorithm. The cable is modeled using Cosserat theory, taking into account conservation of length and weight, as well as bending and torsion. Another method, mainly applied in the context of CG is to model a cable as rigid articulated bodies [96]. Theetten [97] [98] simulates the deformation of One Dimensional (1D) objects with a spline formulation. The method uses continuous expressions for the stretching, bending and twisting energies. A larger number of papers deal with computing the deformation of cables by solving the corresponding differential equations [99] [100] [101]. Pai [99] for example, reduces a cable physical model based on Cosserat theory (he calls it a ‘strand’, a thin elastic object) to a system of spatial ordinary differential equations.

The above describes cases where cables are described as one-dimensional continuous objects. This does not need to be the case: Finite Element Method (FEM) can be used to model cables as well [102] (typically at a higher computational cost than the 1D approaches).

Methods to simulate cables and hoses have been implemented in commercial VR design tools. Dassault Systèmes’ DELMIA product provides a ‘Flex dynamic cable simulation’ [103]. Flexilution offers a package to simulate flexible parts, including wiring harnesses, which is discussed by Goebbels in [104].

The academic and commercial simulations presented above are all aimed at modeling cable/hose/harness behavior for CG or in the 3D design stage. None is aimed specifically at modeling flattening or fitting behavior.

No academic work was found concerning the flexibility of wiring harnesses during (un)flattening. Commercial wiring harness design packages such as CATIA [23], Siemens NX [25] and Pro/Engineer [105] warn their users when a 3D design features
bends with high curvature. A minimum bend radius must be provided by the user beforehand. This is typically a factor of the bundle diameter ($R_{\text{min}} = k \cdot D_{\text{bundle}}$).

**Material properties** The simulations discussed above all require material properties (modulus of elasticity $E$ and the shear modulus $G$). In most cases it is assumed these properties are available or provided by the cable supplier. This may be true for individual wires, but not for wiring bundles. Wiens [106] proposes a method to numerically compute the bending and torsional stiffnesses. The computational results are compared to physical tests. The authors state that the simulation is physically correct, but they indicate as well that other physical issues must be taken into account such as elastic-plastic effects, inner friction and the way of taping. Goss [107] performs experiments to evaluate the material properties of uniform circular rods. His experimental set-up could be used for wiring bundles as well.

**Fokker Elmo approach** Confronted with the issues of predicting the 'flattenability' of a wiring harness, Fokker Elmo experts have developed engineering rules to predict bundle flexibility [16]. These rules have successfully been used to manufacture wiring harnesses for the last decade. The method is therefore considered as reliable and validated for practical wiring harness manufacturing purposes. The approach is proprietary however and can therefore not be detailed here. In short, the analysis method predicts whether a bundle with a certain curvature and other characteristics can or cannot be straightened. A similar method predicts whether a bundle can or cannot be twisted along its axis by a given angle. The main input for these methods is the wiring harness geometry.

### 5.2 Wiring bundle deformation

A formboard engineer needs to determine whether a wiring harness manufactured on a flat table of limited dimensions will fit in the 3D airframe. To determine this he/she evaluates deformations of the flattening (3D to 2D deformation) and fitting (2D deformation) processes are allowed with respect to installation limits. According to wiring manufacturing experts [26] there are two limiting cases for the flexibility of a wiring harness:

- **Damage to the harness**: the damage of contacts or wires.
- **Installation effort**: the force that an installation engineer may physically apply.
Endpoint components and supports are made of metal or hard plastics for example and are treated as fully rigid in evaluating the flexibility of a wiring harness. The wiring harness components that can be considered flexible are in general bundles and covering materials. To evaluate whether the wiring harness can be transformed from its flat manufactured shape into its 3D shape it is required to (1) evaluate the flexibility of the (covered) bundles and (2) determine if a limit has been reached.

**Bundle flexibility** As was detailed in section 2.2, a bundle or bundle-covering combination consists of wire cables of different types, bundled together with tape or tie-wraps and often a combination of covering materials (sleeves, braiding, etc.).

To explore the physics of wiring harness bundles, the mechanics of deformable bodies must be examined. [108] provides the main concepts for the deformations of bodies subject to various types of loading. When considering a wire bundle, an analogy with a deformable rod can be made. The Cosserat theory of rods describes deformable rods that can undergo large deformations in space [109].

**The Cosserat rod** The length of a Cosserat rod is much greater than its width. With small local deformations large displacements can occur at the ends. This behavior is called geometric non-linearity [104]. The rod can in fact be treated as a one-dimensional body that can be described as a curve with length $L$, parameterized by its arc length $s$. Frames along the curve, also called directors, define the material orientation and deformation. The curve and directors are illustrated in the upper part of figure 5.1. The third director $d_3$ is aligned with the tangent of the curve. A Cosserat rod can be subject to the following deformations:

- Stretching, the extension of the rod
- Shearing, the tilting of the cross-section normal
- Twisting, the rotation of the cross-section around its normal vector
- Bending, compression and extension

These deformations are also illustrated in figure 5.1 for a point along the rod. The forces $N(s)$ and moments $M(s)$ on such a point are described by the following expressions [110]:

$$N(s) = N_1d_1 + N_2d_2 + N_3d_3$$
5.2. Wiring bundle deformation

Figure 5.1: Deformation modes of a Cosserat rod

\[ \mathbf{M}(s) = M_1 \mathbf{d}_1 + M_2 \mathbf{d}_2 + M_3 \mathbf{d}_3 \]

Where \( N_1 \) and \( N_2 \) are the shear forces in the \( \mathbf{d}_1 \) and \( \mathbf{d}_2 \) directions, respectively; \( N_3 \) is the axial force; \( M_1 \) and \( M_2 \) are the bending moments in the \( \mathbf{d}_1 \) and \( \mathbf{d}_2 \) directions; and \( M_3 \) is the twisting moment. Including the expressions for the forces and moments we get:

\[ \mathbf{N}(s) = GA\alpha_1 v_1 \mathbf{d}_1 + GA\alpha_2 v_2 \mathbf{d}_2 + EA v_3 \mathbf{d}_3 \]  \hspace{1cm} (5.1)

The shear forces depend on the shearing rigidity \( GA\alpha_{1,2} \) and the shear strains \( v_1 \) and \( v_2 \). The axial force is a function of the axial rigidity \( EA \) and the dilatation.
ν₃. The material properties are set by the modulus of elasticity $E$ (Young's modulus) and the shear modulus of elasticity $G$. $A$ is the cross-sectional area and $\alpha$ is the shear coefficient, which depends on the cross-section.

$$\mathbf{M}(s) = EI_1 (\kappa_1 + \bar{\kappa}_1) \mathbf{d}_1 + EI_2 (\kappa_2 + \bar{\kappa}_2) \mathbf{d}_2 + GJ \tau \mathbf{d}_3$$  \hspace{0.5cm} (5.2)

We see that the bending and twisting moments in a uniform deformable rod depend on the flexural rigidity $EI$ and torsional rigidity $GJ$ and the curvatures ($\kappa_1$, $\kappa_2$ and the twist density $\tau$). Note that in the equation $\bar{\kappa}_1$, $\bar{\kappa}_2$ are the initial curvatures and $I_1$, $I_2$ and $J$ are the moments of inertia about the directors.

In his PhD thesis, Goss [110] provides an overview of the historical and mathematical background of Cosserat rods. For more details on deformable rods refer to works like [109] [111].

**A harness bundle** What are the actual mechanics of wiring harness bundles? A wiring harness bundle will behave in a similar way as a Cosserat rod as for most cases the length of a bundle is much greater than its width. The flexibility of a bundle will be affected by:

- Number of wires
- Types of wires
- Individual wire rigidities
- Cross-section arrangement of wires
- Wire twisting
- Friction between wires
- Distance to break-out or endpoint
- tape/tie-wrap tightness (depending on operator)
- Number and type of covering layers
- Covering properties (material, thickness, braiding type)
- Covering tightness
5.3. Modeling methods selection

Some of these aspects do not vary along the bundle length (e.g. wire type), but some are depending on the position along the bundle (e.g. distance to a break-out). Most bundles with or without covering have some flexibility. In practice, only some very thick bundles and some types of protective materials behave as if they were rigid. At break-out points and endpoints, bundles are typically bound together very tightly and these areas can also be considered nearly rigid, effectively behaving like clamping points. Note that the effective rigidity of a bundle can differ greatly depending on the manufacturing operator. This variety can for example be caused by differences in applied force by operators or the distance between binds. In practice when bending or twisting a bundle it initially deforms elastically. After some time however, the bundle 'sets' itself in the deformed position. This 'plastic' behavior is mostly caused by displacements of bundle components.

5.3 Modeling methods selection

As there is a large body of work available on the simulation of cables, it seems to be an attractive option for modeling wiring harness bundles. However, the approaches presented in the previous section assume that the modulus of elasticity and the shear modulus are known or can be computed. For wiring harness bundles these material properties are not known. To the author’s knowledge, no computational method exists that can reliably predict the flexibility of a bundle and which has been validated by industry. A model for the computation of the material properties would have to be developed, tested and validated. With known material properties, a relation to the harness installation limits (harness damage, installation effort) must be established. This will require further simulations and experimentation. These experiments are costly, especially since new samples will have to be made for each new bundle configuration. No resources have been available during the thesis project to perform these experiments.

The Fokker Elmo analysis methods have been selected for the development of the formboard generation tool. These analysis methods have the disadvantages of being company confidential (and can therefore not be published); and not providing parameters needed to use existing bundle modeling and simulation software. There are important advantages however: the methods have been validated in industry; they relate directly to the damage and installation effort limitations; and match the Knowledge Based Engineering (KBE) objective of reusing company knowledge.
5.4 3D analysis approach

The main analyses that must be performed to determine if the wiring harness can be manufactured in 2D consider: break-out orientations, bundle bending and twisting flexibility. The general approaches to these analyses are explained in this section.

5.4.1 3D break-out detection

A wiring harness break-out can only be flattened if the bundles joining in a break-out point lie in a common plane. The splitting bundles direction vectors are only allowed to deviate from this common plane by a small amount. As indicated in section 2.4, a break-out point should be designed in a single plane in order to enable manufacturing in a flat plane. Unfortunately the DMU geometry does not always allow this, or the 3D designer (from another company) may not be aware of this requirement. A design can therefore feature so-called 3D break-outs, as shown schematically in the right of figure 5.2. These are break-outs where the bundles are not positioned in a single plane. In case a 3D break-out is identified, it must be re-designed (preferably) or if this is not possible (semi)3D tooling is required. In most cases bundles are not exactly positioned in the same plane at a break-out point. For these cases the break-out point can often be deformed to a certain degree to allow manufacturing in a flat plane. Such a case, illustrated in the center of figure 5.2 requires analysis of the break-out to determine if flat manufacturing is possible.

![2D break-out](image1)

![3D break-out?](image2)

![3D break-out](image3)

Figure 5.2: Schematic representation of three break-outs. Left: bundles are in common plane, and can therefore be manufactured in a flat plane. Center: bundles are not in a common plane, the manufacturability must be evaluated. Right: bundles are not in a common plane and cannot be manufactured on a flat plane.

Presently at Fokker Elmo, the 3D digital model of the wiring harness is visually examined by experienced manufacturing engineers to identify 3D break-outs. Typically, a manufacturing engineer will navigate through the 3D harness model and
visually inspect 'suspicious' break-outs and decide whether the first few centimeters of splitting bundles are sufficiently positioned in a single plane to allow flattening. Break-outs tend to be more 'suspicious' when featuring particularly thick bundles. The evaluation by the manufacturing engineer is based on tacit knowledge that must be made explicit in order to automate the task. The approach that follows next did not formally exist and was developed based on the available manufacturing engineering experience.

Figure 5.3: Trimetric view of a break-out point with four bundles indicating vector definitions and the bundle flattening angle.

The flattening plane or break-out plane is the plane a break-out will be manufactured in. It is defined by the 3D plane vector $P$ and the connection point center $C$ as indicated in figure 5.3. For each bundle at a break-out point, the rule for the break-out analysis is that the angle $\theta_i$ between the 3D bundle direction vector $B_i$ and its projection on the break-out plane $F_i$ remains lower than a limit angle $\theta_{limit}$, or: $\angle(B_iF_i) = \theta_i < \theta_{limit}$, where $F_i \perp P$. Before this evaluation can be performed, the following steps must be performed:

1. Determine the 3D bundle direction vector $B$ for each bundle at the break-out point.
2. Determine the break-out plane vector $P$ of the break-out point.
3. Project the 3D bundle direction vector $B$ to the break-out plane in order to obtain the flat bundle direction vector $F$.

1. **Bundle direction determination** The 3D bundle direction vector $B_i$ denotes the direction of bundle $i$ at the connection point $C$. One candidate is the tangent vector of the bundle centerline $T_i$, as illustrated in figure 5.4. This vector is not always representative however as centerline tangents of different bundles are often
aligned at the connection point before breaking out. Therefore a reference vector \( R_i \) is computed by determining a point \( P_{\text{ref},i} \) along the bundle centerline at a reference distance \( d_{\text{ref}} \) and computing the vector between these points (i.e. vectorial subtraction of the points: \( R_i = P_{\text{ref},i} - C \)), which is illustrated in figure 5.4.

Figure 5.4: Candidates for the bundle direction vector. Left: The centerline tangent at the connection point. Right: A reference vector between the connection point and a reference point at arc length \( d_{\text{ref}} \).

Which of the two vectors is selected as the 3D bundle direction vector \( B \) depends on the vectors and bundle thickness of the other bundles of the break-out point. The tangent vector \( T \) is used when:

- The bundle has the largest diameter of all bundles at the break-out point \( T_{D_{\text{max}}} \);
- The bundle has the largest diameter of all bundles with tangent vector \( T \) exactly opposite of \( T_{D_{\text{max}}} \);
- The bundle tangent \( T \) is not parallel to \( T_{D_{\text{max}}} \).

In all other cases, the reference vector \( R \) is used. Figure 5.5 shows an example case where these rules have been applied.

2. Break-out plane determination The bundle direction vectors are used to determine the break-out plane vector \( P \). The objective is to obtain a break-out plane that results in the smallest bundle flattening angle \( \theta \), see figure 5.3. The bundle with the largest diameter (i.e. the main bundle) is positioned in the break-out plane by definition (i.e. \( \theta_{\text{main}} = 0 \)). The bundle direction vector of the main bundle \( B_{\text{main}} \) is used as a reference vector to compute a plane vector \( P_i \) for each other bundle at
5.4. 3D analysis approach

Figure 5.5: Example of the selection of bundle direction vectors.

the connection point: $\mathbf{P}_1 = \mathbf{B}_{\text{main}} \times \mathbf{B}_i$. No plane vector is computed when a bundle direction vector $\mathbf{B}_i$ is parallel to $\mathbf{B}_{\text{main}}$.

Figure 5.6: Plane vectors are computed from the bundle direction vectors determined in figure 5.5. At the left side the resulting plane vector $\mathbf{P}$ is indicated.

In case only a single plane vector $\mathbf{P}_1$ is computed, $\mathbf{P} = \mathbf{P}_1$. When more than one vector $\mathbf{P}_i$ is computed, such as in figure 5.6, the break-out plane vector $\mathbf{P}$ is found by computing the vector yielding the smallest bundle flattening angle $\theta_i$. This can be done by computing the angles between one plane vector $\mathbf{P}_0$ and other plane vectors $\mathbf{P}_1, \mathbf{P}_2, ...$ yielding a set of angles $\delta_1, \delta_2, ...$. The mean of the minimum and maximum angles $\delta_{\text{min}}$ and $\delta_{\text{max}}$ gives the angle $\theta_0$ between $\mathbf{P}_0$ and $\mathbf{P}$. Rotating plane vector $\mathbf{P}_0$ around $\mathbf{B}_{\text{main}}$ gives the break-out plane vector $\mathbf{P}$. Figure 5.7 shows a few examples where the break-out plane vector is determined from bundle direction vectors.
5. 3D ANALYSIS AND FLATTENING OF A WIRING HARNESS

3. Bundle projection  In the manufacturing state, bundles are positioned in the break-out plane. The corresponding flat bundle direction vectors $F_i$ are obtained by rotating the bundle direction vector $B_i$ with the bundle flattening angle $\theta_i$ around the vector given by $B_i \times P$. The flat bundle direction vectors are indicated in figure 5.8.

Figure 5.8: The bundle direction vectors of figure 5.5 are rotated to the break-out plane in order to obtain the flat bundle direction vectors $F_i$. 
4. **Identification** A break-out can be flattened when each bundle flattening angle is smaller than a limit angle: \( \theta_i < \theta_{limit} \). Presently the same limit angle (e.g. 5 degrees) is implements for all break-outs, irrespective of bundle properties. A break-out where a bundle flattening angle violates the limit is designated as a 3D break-out and presented to an expert. This expert is given a 3D view of the break-out and the properties of the joining bundles, based on which he/she can decide to overrule the designation of the break-out as not flattenable. These decisions could be recorded to provide the basis for a future set of rules that are a function of break-out bundle attributes.

5.4.2 **Bundle flexibility analysis**

To reshape a wiring harness from the 2D manufacturing state to 3D installation state it must be bent and twisted to the desired form. Axial and shear forces (equation (5.1)) do not allow large deformations of a bundle and are therefore not applied to reshape a wiring harness. The required deformations from 2D to 3D are achieved by applying moments to a bundle. Whether it is allowed to flatten a bundle section depends on the bundle flexibility and corresponding deformation limits. The parameters determining the flexibility of a uniform bundle section are the rigidities and the curvatures as shown in equation (5.2).

**Twisting analysis** Consider manufacturing the wiring harness on a perfect physical 3D mock-up. The wires in the bundles would be manufactured in their installation state, their rest state. These wires would follow a 3D path from one connection point to the next, for example from connection point \( C_s \) to \( C_e \) in figure 5.9, each having a plane vector \( (P_s, P_e) \) respectively as defined in section 5.4.1. In practice the manufacturing state is preferably 2D, which means that the plane vectors \( P_s \) and \( P_e \) must be aligned with each other in order that both break-out planes correspond to the formboard plane. To evaluate whether this is possible the required twisting angle \( \theta_{twist} \) needs to be determined and compared to a flexibility or deformation limit.

A 3D path with zero twist angle (\( \theta_{twist} = 0 \)) can be modeled from one connection point to the next. This is illustrated in figure 5.10 where the start plane vector \( P_s \) is transported along the bundle section centerline from connection point \( C_s \) to connection point \( C_e \) by means of a minimum rotation sweep. This transported vector is the zero twist plane vector \( P_{e,0} \). The angle difference between the axis defined by \( P_{e,0} \) and \( C_e \) and the end plane vector \( P_e \) is the minimal twist angle needed for flattening: \( \theta_{twist, min} = min(\angle P_{e,0} P_e; \angle P_{e,0} P_e) \).
Figure 5.9: Schematic bundle section with known start and end plane vectors $P_s$ and $P_e$. The twisting analysis needs to evaluate whether the bundle section is sufficiently flexible to align both plane vectors.

Figure 5.10: The minimum twist angle required to align plane vectors $P_s$ and $P_e$ is the angle between the zero twist end plane vector $P_{e,0}$ and $P_e$. 
The minimum twist angle $\theta_{\text{twist}, \text{min}}$ can be compared to the allowed twist $\theta_{\text{twist, allowed}}$ considering the bundle rigidity parameters. Rigidity parameters are determined based on bundle attributes such as its length and diameter. Two methods are available to compare the minimum twist angle: (1) an experiment-based proprietary method by Fokker Elmo and (2) a maximum allowed twist angle per unit length value: $\frac{\theta_{\text{twist, min}}}{L_{\text{bundle}}} < \theta_{\text{twist, allowed}}$. The latter can be made dependent on bundle attributes such as the material type. Figure 5.11 shows the activity diagram of the twisting analysis.

Three methods to compute RMFs are: the projection method by Klok [112], the rotation method by Bloomenthal [113] and the double reflection method by Wang [114]. The double reflection method (with a fourth order global approximation error) is shown by Wang [114] to be more accurate than the other two methods (with a second order approximation error) and is therefore selected. The algorithm is given in figure 5.13.

---

1 This was verified by comparing results of the presented approach with a Fokker Elmo case.
Bending analysis  The bundle curvature distribution is analyzed for violations of minimum bend radius limits. These indicate how far a straight section can be bent, i.e. the minimum bend radius applies to the comparison of a bent section to a straight section. It is desirable to obtain straight sections from flattening, in the first place for manufacturing reasons as straight bundle sections require less formboard preparation time and have ergonomic advantages. A flattened wiring harness with straightened sections also provide a clearer overview to perform the fitting step.

The curvature distribution is determined by evaluating the curvature $\kappa_i$ at each point $x_i$ (where $i = 0, 1, \ldots, n$ with arc lengths $s(x_0) = 0$ and $s(x_n) = L_{\text{bundle}}$, see figure
5.12) along a bundle centerline. The curvature evaluation is performed with the CAD kernel \(^2\) curvature at a specified curve parameter function for NURBS curves. Figure 5.14 shows a schematic illustration of the curvature distribution versus the bundle length.

The minimum bend radius \(R_{\text{min}}\) can be evaluated through two methods: (1) an experiment-based proprietary method by Fokker Elmo and (2) a multiplication factor of the bundle diameter \(R_{\text{min}} = nD_{\text{bundle}}\). In both cases certain flexural rigidity parameters are required such as the bundle diameter and several other bundle attributes. These are used for example by method (2) where the multiplication factor can be made a function of the material type (i.e. \(n = f(\text{material-type}, D_{\text{bundle}})\)).

The minimum bend radius can be translated in a curvature limit \((\kappa_{\text{limit}} = \frac{1}{R_{\text{min}}}\) allowing segments to be identified in the curvature distribution that violate this limit (see center of figure 5.14). A filter is applied to eliminate segments that are too short with respect to a threshold value to be relevant for the analysis (e.g. segments of less than 1 mm length). Segments that do not violate the curvature limit can be straightened, segments that do need further analysis.

A further test determines whether the bend can be reproduced in a plane without violating the curvature constraints, i.e. whether sections that cannot be straightened are allowed to be flattened. This is done by defining a flattening plane for the bundle segment and evaluating if the curvature has a component out of this that violates the curvature limit \(\kappa_{\text{limit}}\). In other words, determine whether the flat-bent section can be bent into the 3D shape. As indicated in figure 5.14, a negative outcome of this analysis means flattening the harness is not allowed. The procedure to evaluate the flattenability is as follows:

Along the centerline Frenet frames are computed for each \(x_i\) position along the bundle segment centerline, see figure 5.15. A Frenet frame consists of the curve tangent vector \(T_i\), the normal vector \(N_i\) giving the curvature direction and the binormal vector \(B_i\) \((B_i = N_i \times T_i)\). A flattening plane \(P_{\text{flat}}\) is defined for the bundle segment, for example by computing the cross product of the segment start and end tangent vectors \((P_{\text{flat}} = T_0 \times T_n)\). This flattening plane vector is propagated along the centerline using RMFs, as illustrated in figure 5.15. Each plane vector \(P_i\) represents the vector pointing out of the formboard plane in the 3D state.

The Frenet normals \(N_i\) and the plane vectors \(P_i\) allow evaluating the component of the curvature out of the formboard plane. In case the bundle segment would

\(^2\)GDL uses the SMLib CAD kernel [115], based on [116].
5. 3D ANALYSIS AND FLATTENING OF A WIRING HARNESS

Figure 5.14: Schematic overview of the wiring harness bending analysis steps

Figure 5.15: Determination of reference frames along a bundle segment centerline. Left: Frenet frames are computed for each $x_i$. Center: A flattening plane $P_{flat}$ is selected. Right: RMF are generated to transport $P_{flat}$ from $x_0$ to $x_n$. 

104
be oriented in a single plane already, the plane normals $P_i$ would be perpendicular to the Frenet normal vectors $N_i$. This is not the case in the example of figure 5.16. At each point $x_i$, the normal vector $N_i$ can be decomposed in a component parallel to the plane vector $P_i$ and a component perpendicular to it ($N_{i\text{out}}$ and $N_{i\text{in}}$, respectively). The magnitude of the out-of-plane curvature can be computed with $||\kappa_{i\text{,out}}|| = ||\kappa_i|| \frac{||N_{i\text{out}}||}{||N_i||}$. The distribution of the out-of-plane curvature can be evaluated with one of the curvature analysis methods as indicated in the lower part of figure 5.14.

If the wiring harness does not comply to the minimum bend radius constraint, the user must be informed that flattening is not allowed. When this is the case there are two options: redesign the 3D model in order to ensure it can be flattened or use costly 3D tooling (physical mock-ups). In case no violations are present the model can be flattened, according to the procedure described in the following section.
5. 3D ANALYSIS AND FLATTENING OF A WIRING HARNESS

5.5 Flattening approach

When it has been verified that no 3D break-outs (section 5.4.1) or violations of flexibility limits (section 5.4.2) are present in the wiring harness model, the transformation from 3D to 2D can take place.

5.5.1 Existing methods

As introduced in section 2.4, the standard commercial Computer Aided Design (CAD) tool used for flattening wiring harnesses is electrical wiring workbench of CATIA V5 [14]. This package, as well as some other CAD packages, automatically flattens wiring harnesses using a projection algorithm. The problems associated with this approach were illustrated in figures 2.11, 2.12 and 2.13.

Mentor Graphics uses an 'unfolding method' [117] (similar to [118]) that discretizes bundles into straight and curved segments. The approach sequentially transforms (i.e. 'unfolds') the curved segments by straightening them and rotates segments to the target plane. Branches are rotated to the formboard plane either around the main bundle segment or around the branch-plane vector cross-product. Like CATIA, the method selects a flattening plane, but does so in a smarter way by selecting it based on the harness geometry. Because the method depends on the selection of a starting (main) branch, a fixed flattening plane and a discretization of bundle geometry, the results can lead to similar issues as experienced with the projection method.

To achieve a reliable flattening approach, the following two options have been considered:

1. Use the flattening methods provided by current CAD systems and automate all the required manual checks and adjustments.
2. Develop an alternative flattening method that eliminates the unnecessary checks and adjustments and automates the required ones.

Option 2 would require the development of a new approach to wiring harness flattening. Option 1 would maintain the use of an approach already familiar to wiring manufacturing engineers, which is, however, the main cause of problems. Effectively option 1 is a 'patch' and option 2 tackles the root of the problem. Eventually, option 2 was selected, because the availability of industrial wiring harness manufacturing and installation experts was the most favorable condition to initiate the development of an alternative method.
5.5. Flattening approach

5.5.2 Proposed alternative

The most important attribute values for flattening are actually already determined by the 3D analysis discussed in section 5.4. In order to obtain a flat wiring harness the following tasks, schematically illustrated in figure 5.17, are performed [119]:

- A. Connection points must be flattened
- B. Bundles must be flattened
- C. Bundles must be adapted to rotate all connected components onto a single plane

A. Connection point flattening  The 2D parameters of connection points connecting multiple bundles (i.e. break-outs) are determined by rotating the bundle direction vectors \( \mathbf{B}_i \) in figure 5.8 to the break-out plane as shown in section 5.4.1. The resulting flat bundle direction vectors \( \mathbf{F}_i \) are defined in the global coordinate system. In order to obtain an independent parameterization, the angles \( \beta_i \) between the flat bundle direction vectors \( \mathbf{F}_i \) and a reference vector \( \mathbf{F}_{\text{ref}} \) around the break-out plane vector \( \mathbf{P} \) are computed. The main bundle direction vector is used as the reference vector \( \mathbf{F}_{\text{ref}} = \mathbf{F}_\text{main} \). These angles are illustrated in the center of figure 5.18. In practice it is desired to round the break-out angles to certain step sizes (e.g. 5 degrees). Fanning rules are implemented to ensure a minimum angle between break-out angles (e.g. \( \geq 10 \) degrees). The result of rounding and fanning is shown in the right part of figure 5.18. With the break-out angles \( \beta_i \) (or \( \beta_{R,i} \)) and the reference bundle (bundle for which \( \beta_i = 0 \)) the break-out can be reconstructed in the flat state.

Since endpoint components are rigid, their geometry does not change, i.e. no 2D parameters have to be computed. Their orientation in the plane entirely depends on the relative orientation to the connection point plane normal vector \( \mathbf{P} \) linking them to a bundle. This vector is selected from a set of allowed planes \( \mathbf{P}_{\text{allowed},i} \) the endpoint components can be positioned in. Theses allowed planes are set by rules that are specific to an endpoint instance geometry. They are defined as angles \( \alpha_i \) relative to the endpoint normal angle \( \mathbf{N}_{\text{endpoint}} \) and tangent \( \mathbf{T}_{\text{endpoint}} \). For example, the presence of an angled backshell will lead to an allowed plane vector \( \mathbf{P}_{\text{allowed}} \) corresponding to the cross-product of the backshell centerline vectors. Some components do not require a specific flattening plane (e.g. a circular connector). This is illustrated in figure 5.19.
5. 3D ANALYSIS AND FLATTENING OF A WIRING HARNESS

Figure 5.17: Schematic illustration of the procedure to generate a 2D wiring harness representation. Each connection point is flattened (A), each bundle is flattened (B) and a twist angle is applied to each bundle (C).

Figure 5.18: The bundle direction angles $\beta_i$ are defined with respect to a reference bundle direction vector $F_{\text{ref}} = F_1$. In practice these angles are rounded and fanned as well, giving rounded angles $\beta_{R,i}$. 
5.5. Flattening approach

B. Bundle flattening

When allowed, a flat bundle representation is straightened as illustrated in figure 5.17b. The length of the flat bundle centerline is identical to the length of the 3D bundle centerline.

When reconstructing a bend to be respected manually, a formboard engineer measures the (circular) radius of curvature from three points on the bundle centerline. In the flat, straightened wiring harness model, a circular bend with the measured radius of curvature is applied to the bundle section in question. A similar approach is taken for the automated solution. The main difference is that the bundle section is discretized into small segments in order to obtain a bend that more closely matches the original 3D centerline compared to a single circular arc. The approach taken is as follows.

When a bend must be respected the flattening plane vector $P_{\text{flat}}$ of the section is determined as described in section 5.4.2. The section is divided in equidistant points $x_i$ at which Frenet frames and RMFs are defined as in figure 5.16. For each point $x_i$ the in-plane component of the Frenet normal vector $(N_i)_i$ can be computed. This allows computing the in-plane component of the curvature $||\kappa_{i,i}|| = ||\kappa_{i,i}|| \frac{||(N_i)_i||}{||N_i||}$. The flat state of the bundle section is created by constructing circular arc segments of length $ds_i$ and radius $R_i$, where $R_i = \frac{1}{\kappa_{i,i}}$ and connecting these to form a single curve. Note that the arc length is $ds$ for points $x_2$ to $x_{n-1}$ and $\frac{1}{2} ds$ for points $x_1$ and $x_n$. Figure 5.20 shows the resulting curve and also illustrates the difference with using a projection of the 3D section.

Figure 5.19: For each endpoint component, the allowed flattening plane(s) are determined based on component specific rules.
C. Achieving minimum twist  

In order to generate the 2D state of the wiring harness model, a rotation is applied to align successive connection planes as illustrated in figure 5.17c. The rotation angle is the minimal twist angle $\theta_{\text{twist, min}}$, as indicated in figure 5.10. In case a flattened bend is present, the minimum twist is computed in the same manner; the only difference is that the plane vector will be the bundle section flattening plane $P_{\text{flat}}$ (see figure 5.15) instead of the plane vector $P_1$ of a connection point.

The determination of connection point plane vectors $P$ was explained for breakouts in section 5.4.1. Figure 5.19 illustrates that the plane vectors for endpoint connection points depend on the allowed plane vectors $P_{\text{allowed}}$ of the connecting endpoint. For two cases in this figure a definitive connection point plane vector $P$ is not yet selected. This selection is shown in figure 5.21. Vector $P_1$ is transported to connection point $C_3$ ($P_{3, \text{zerotwist}}$) where its orientation is compared to the allowed vectors $P_{3, \text{allowed}(1)}$ and $P_{3, \text{allowed}(2)}$. The allowed vector with the smallest angle to the axis defined by $P_{3, \text{zerotwist}}$ is selected as the connection point plane vector $P_3$. In the case of connection point $C_6$ no allowed angles are specified. The connection point plane vector $P_6$ is set equal to $P_{6, \text{zerotwist}}$, which is $P_2$ transported from $C_2$ to $C_6$. With a connection point plane vector $P$ defined for each connection point, the flattened model can be constructed.

All the parameter values required to generate the flat state of the wiring harness are already computed during the 3D analysis. The flat state is built up from a start connection point. Any can be selected, it has no effect on the outcome except for the visualization on screen. Figure 5.22 illustrates how components (bundles, endpoints) are connected to this point and the follow-up steps leading to the flattened harness.

The result of these steps is a fully flat wiring harness where only minimal rotation
needs to be applied to bundles in order to obtain the 3D orientation. The approach is independent of a projection plane, no matter at which connection point flattening is started, the same result will be produced. This approach differentiates itself from existing methods because:

- It ensures flat break-out directions are as close to the corresponding 3D directions as possible;
- It respects bends that cannot be straightened;
- It guarantees that the minimum of twist needs to be applied to transform the flat harness to its 3D shape;
- It is independent of a projection plane;
- It is independent of a starting bundle.
5. 3D ANALYSIS AND FLATTENING OF A WIRING HARNESS

Analyses are performed on the 3D wiring harness model.

Select a plane, to represent the flat harness in, for example the X-Y plane.

Select any break-out point (e.g. \( C_1 \)) and define its flat geometry at \((0\ 0\ 0)\) with a flat reference tangent and plane vector.

Rotate the reference tangent \( \mathbf{T}_{1,\text{ref}} \) around \( P_1, \text{flat} \) with the bundle direction angles \( \beta_{1,j} \).

Define the endpoint tangent \( \mathbf{T}_{1} \) opposite to \( \mathbf{T}_{1,\text{flat}} \), flat.

Generate the flat geometries of the components connected to \( C_2, C_3 \) and \( C_4 \).

Apply the minimum twist angle to align the connection point plane vectors.

Rotate the reference tangents \( \mathbf{T}_{i,\text{ref}} \) around \( P_i, \text{flat} \) with the bundle direction angles \( \beta_{i,j} \). Define the endpoint tangent \( \mathbf{T}_{i} \) opposite to \( \mathbf{T}_{i,\text{flat}} \).

Generate the flat geometries of the components connected to \( C_5 \) and \( C_6 \).

Apply the minimum twist angle to align the connection point plane vectors. Define the endpoint tangent \( \mathbf{T}_{6} \) opposite to \( \mathbf{T}_{6,\text{flat}} \).

Generate the flat geometries of the components connected to \( C_7 \) and \( C_8 \).

Figure 5.22: Schematic illustration of the steps taken to generate the flat state of a wiring harness.
The analysis and flattening approaches are implemented in a KBE application as detailed in the next section.

5.6 KBE implementation

The 3D analysis and flattening approach presented in the previous section have been implemented in a KBE application named f3flat using the GDL KBE system [53]. Figure 5.23 shows the general Unified Modeling Language (UML) activity diagram of the application. The 3D wiring harness design is imported, its components are sorted and connections between elements are defined. Wiring harness High Level Primitive (HLP)s (discussed below) are then instantiated, initially only showing 3D geometry. The various 3D analyses are then performed, followed by flattening steps. Finally the flat geometry can be visualized and a Capability Module (CM) enables exporting it. The steps of this activity diagram are further detailed in this section, after briefly discussing the f3flat product model structure.

The f3flat application class is wiring-harness, as shown in figure 5.24. Note that the class diagram uses a GDL-specific UML notation: the aggregate classes of wiring-harness are child-classes which are generalizations (types) of a HLP class. Refer to appendix G for more details on this UML notation. The HLP classes are: connection-point, bundle, endpoint, covering and attached-component. The wiring-harness class inherits the properties of the General-purpose Declarative Language (GDL) base-object class as well as a CM that collects the wiring harness output definition. The dependency arrows indicate that the child-classes have (inter)dependent slots, defined in wiring-harness. The bundles child-class for example provide geometric positioning to the coverings child-class. The latter again provides flexibility parameters to bundles. More detailed class diagrams of the f3flat HLPs are given in appendix H.

Import 3D harness definition A standard input format is defined for each HLP, based on the relevant product model features for this formboard design step. A bundle for example requires the NURBS definition of its centerline; figure 1.1 gives an overview of the inputs. The input is currently stored in a simple .dat file, but it could as well be stored in XML format or in the company database. The latter was not yet possible. The creation of the input from wiring harness CAD models is further discussed in chapter 7. Given the path to an input file, the wiring-harness class extracts the required data.

Sort bundles & define connections To initialize the model, the bundles input definition is sorted using the sorted-bundles class: a tree graph is constructed based
on the connections between bundles. Based on this graph connection points are defined at each break-out and endpoint. These connections act as interfaces between bundles and other bundles or endpoints (i.e. they manage the interrelations). A bundle will be connected to a connection point, instead of another bundle. The input data enriched with connectivity attributes is stored as the \texttt{sorted-bundles} computed slot.

**Generate 3D model geometry** The input data provides the geometry parameter values needed to generate the \texttt{bundle} and \texttt{endpoint} classes 3D geometry (e.g. NURBS definition of the bundle centerline). These are generated and provide reference geometry in order to instantiate covering, \texttt{*-attached-component} and
connection-point geometries, see figure 5.25. The geometry of the latter is a 3D point, positioned at the extremity of the connections reference bundle.

**Perform break-out analysis**  The analysis of break-outs is performed within the connection-point class, based on centerlines provided by the bundle class as shown in figure 5.26. For each connected bundle, its tangent vector $\mathbf{T}_i$ at the connection point is computed as well as reference vectors $\mathbf{R}_i$ as illustrated in figure 5.4. From these vectors, bundle direction vectors $\mathbf{B}_i$ are selected. The procedure for selecting these reference vectors is shown in the right part of figure 5.26. With these vectors the break-out plane vector $\mathbf{P}$ yielding the smallest bundle flattening angles $\theta_i$ can be computed, as shown in figure 5.6. These angles are then compared to the 3D break-out limit angle and a warning is given to the user in case the limit is violated. The user is warned for example by the coloring (in red) of the break-out plane in the 3D model.
5. 3D ANALYSIS AND FLATTENING OF A WIRING HARNESS

![Activity diagram of the generate 3D model geometry activity of figure 5.23.]

Figure 5.25: Activity diagram of the generate 3D model geometry activity of figure 5.23.

![Activity diagram of the perform break-out analysis activity of figure 5.23.]

Figure 5.26: Activity diagram of the perform break-out analysis activity of figure 5.23.
Figure 5.27 shows screenshots from the \textit{f3flat} application where break-out planes are represented by discs. These are either light gray or red, where the last indicates the 3D break-out limit has been violated.

![Illustration of flattenable and 3D break-outs, detected by the \textit{f3flat} application.](image1)

Figure 5.27: Illustration of flattenable and 3D break-outs, detected by the \textit{f3flat} application.

Figure 5.28: Illustration of bending analysis results, generated by the \textit{f3flat} application.

Figure 5.29: Illustration of twisting analysis results, generated by the \textit{f3flat} application.
Perform bending analysis The bending analysis is performed within the bundle class and is independent of other HLP instances, except for covering. The activity diagram is given in figure 5.30. It starts by generating the bundle centerlines and computing the bundle length. Using amongst others flexibility parameters provided by covering instances, the bundle is divided into segments of uniform stiffness. For each of these segments uniform-bundle-segment classes are instantiated that evaluate the curvature distribution and perform the bending analysis as described in section 5.4.2. Either the Fokker Elmo or diameter multiplication method \( R_{min} = nD_{bundle} \) can be used, for which corresponding CMs are available. The uniform-bundle-segment instances provide lists of straightenable and non-straightenable sections that are collected into a single list at the bundle level.

Bundle sections are generated and for each non-straightenable bundle-section instance, the flattenability analysis is performed. This starts by computing the section flattening plane \( P_{flat} \) and the out-of-plane curvature distribution as explained in section 5.4.2. One of the bending analysis methods will determine whether the section can be flattened. If this is not possible, a warning is given to the user. The different analysis results are illustrated in figure 5.28, which shows screenshots of the f3flat application bending analysis results.

Perform twisting analysis The principles of the twisting analysis were presented in section 5.4.2. Its implementation encompasses a number of classes, in particular the bundle class and its subclasses. The break-out connection point provides the connection plane normal \( P \), after which bundle-section instances are generated within the bundle class. Figure 5.31 shows the activity diagram for the twisting analysis from the perspective of a single bundle-section. The analysis is based on straightenable bundle sections, in case it is not, the plane normal of the bend section \( P_{flat} \) is evaluated as explained in the previous paragraph.

When the section is straightenable the maximum allowed twist \( \theta_{twist,allowed} \) is computed using one of the two methods indicated in figure 5.11. The zero twist end plane vector \( P_{e,0} \) at the other end of the section can be computed (see figure 5.10). This vector must be compared to the plane vector \( P_e \) at that point. There are three cases. (1) The succeeding section is bent therefore its plane vector \( P_{flat} \) is selected. (2) The succeeding section is straight, the plane vector \( P_e \) is set equal to the zero twist plane vector \( P_{e,0} \). (3) The section ends in an endpoint connection, the endpoint plane normal \( P_E \) is computed. This is done by determining the free endpoint plane vector (zero twist) in bundle and evaluating the allowed planes \( P_{allowed,i} \) for the endpoint in endpoint. The connecting connection-point compares these vectors and selects the connection plane normal with the least amount
5.6. KBE implementation

Figure 5.30: Activity diagram of the bending analysis activity of figure 5.23
5. 3D ANALYSIS AND FLATTENING OF A WIRING HARNESS

Figure 5.31: Activity diagram of the twisting analysis activity of figure 5.23, from the perspective of a single bundle-section.
The angle between the the zero twist vector $\mathbf{P}_{e,0}$ and the plane vector $\mathbf{P}_e$ is computed for each bundle-section. These angles are distributed evenly in terms of twisting capacity over succeeding straightenable sections within bundle. This resulting effective twist $\theta_{\text{effective}}$ is assigned to each section where it can be compared to the twisting limit $\theta_{\text{twist,allowed}}$. When the limit is violated the user is warned, if not the effective twist for a flipped bundle section $\theta_{\text{flipped}}$ is evaluated. Flipping a bundle means reversing the plane vector of one end with respect to the other. This means a larger twisting angle is applied to the bundle section. This angle is compared to the allowed twist and a warning is provided in case flipping is not allowed. Note that the last case does not mean that the harness is not flattenable.

The different analysis result cases are shown in figure 5.29 for two wiring harnesses. The analysis feedback colors are green for sections within limits that can be flipped; blue for sections within limits that cannot be flipped; orange for sections with lengths falling outside the analysis range; and red for sections where the twisting limit is violated for flattening.

**Flatten connection points** The 3D analysis provides the 3D connection point plane vector $\mathbf{P}$ and bundle direction vectors $\mathbf{B}_i$ (see figure 5.3). As indicated in figure 5.32 the bundle direction vectors are projected (i.e. rotated with the minimum angle) to the 3D plane resulting in flat bundle direction vectors $\mathbf{F}_i$. Next the angles $\beta_i$ between the flat bundle direction vectors and a reference vector is computed, with respect to the 3D plane vector (see figure 5.18). Depending on the aircraft project some rounding and fanning rules can be applied, for example rounding to five numbers and fanning with a minimum angle difference of 15 degrees.

**Flatten bundles** Each bundle-section of a bundle is analyzed for bending and when the section is straightenable a straight centerline can be generated as shown in figure 5.33. When the section is not straightenable, the in-plane curvature is computed and a bend centerline is generated by creating small arc sections matching the in-plane curvature that are combined into a single curve as detailed in section 5.5.2.

**Evaluate and apply rotation** The relative position of bundle-sections is determined by the normal vectors at their start and end points. For straightenable sections, the twisting analysis provides the effective twist, which is used to set the normal vectors. For a flattened bend, the normal vectors are set according to the bend.
Figure 5.32: Activity diagram of the connection point flattening activity of figure 5.23.

plane. With these computations, all parameters describing the flat configuration are present, all that remains is to visualize the 2D model.

Figure 5.35 shows the effect of changing the flexibility parameters (in this case the diameter) on the 3D analysis and flattening of a bundle.

**Visualize 2D model, generate output**  For the 3D model the bundle and endpoint components position and orientation were provided by the input and the connection-point position was determined by a connecting bundle. For the 2D model however, connection-point objects determine the position and orientation of bundle and endpoint components (see figures 5.22 and 5.34). By default, the starting flat connection-point is positioned at $C_{1,\text{flat}} = (000)$ with plane vector $P_{1,\text{flat}} = (001)$ and start tangent $T_{1,\text{flat}} = (100)$ (see figure 5.22). Another 3D point or vectors could also be used without any consequences to the flattened model. Note that the flat connection-point planes are constrained to be either the positive or the
5.6. KBE implementation

![Activity diagram of the bundle flattening and evaluate and apply rotation activities of figure 5.23.]

negative z-direction. This removes round-off errors which may lead to divergence from the flat plane.

The connection-point provides its position and plane vector to the connected bundle objects. The bundle tangent vector $T_{i(j)}$ is then computed by rotating the connection point tangent $T_{i,\text{flat}}$ with the bundle direction angle $\beta_{i(j)}$ around the plane vector $P_{i,\text{flat}}$. The flattened bundle centerline is positioned according to these point and vectors.

At the end of each bundle the flat state of the connecting connection point is defined by a position, tangent vector and plane vector: (001) or (00 − 1), depending on the twisting of the bundle. When the connection-point is a break-out, bundles are positioned as indicated by the loop in figure 5.34. When the connection-point connects to an endpoint, the orientation of the endpoint with respect to the plane is determined and the endpoint component geometry is generated accordingly, see figure 5.22.

The covering and attached component geometries are generated again with respect to the flattened bundle or endpoint components. Two example wiring harnesses that are flattened are shown in figure 5.36.
Figure 5.34: Activity diagram of the *visualize 2D model* activity of figure 5.23.

Figure 5.35: Illustration of the effect of bundle diameter on bending analysis results for a single bundle, generated by the *f3flat* application.
A CM has been implemented to store a flattened wiring harness definition. It consists of definition-collection-* classes corresponding to each model primitive that extract the relevant parameters. Stored connection-point parameters are the index, connecting endpoint index, connecting bundle indices, bundle direction angles and a boolean to indicate whether the connection point is located at the bundle start. The bundle parameters that are stored are the id-string, index, length, diameter, connection-point index and the definition of flattened bends. The f3flat output file format is shown in figure I.4.

The f3flat application This section has shown how the approaches presented in sections 5.4 and 5.5 were implemented in a KBE application. The implementation has been made accessible through a Graphical User Interface (GUI), illustrated in figure 5.37. The GUI features the wiring harness product tree, a viewer to display geometry, an inspector to inspect component attributes and a menu to control the interface viewing modes.

5.7 Discussion: is this design?

Existing engineering rules have been applied and new approaches have been devised to perform the laborious task of analyzing and flattening a wiring harness 3D model. The resulting f3flat KBE application fully automates all steps, leaving the formboard engineer only the task of assessing analysis violations.

The evaluation of (bundle) flexibility is done by means of experiment-based rules, validated by years of industrial practice at Fokker Elmo. Higher fidelity, physics-based approaches can be implemented when the corresponding predictions of bundle flexibility and its effect on human installation effort and handling damage are available and are at least as reliable as the current methods.

In the 'old' formboard design process, engineers were creating different flattened configurations depending for example on experience, perspective of the 3D model and analysis thoroughness. The corrections that were iteratively introduced in the flattened model caused flattening to be considered as a design activity. The development of the new flattening approach shows that with the available rules, the problem is fully constrained: there is only one single best configuration to be created from a 3D wiring harness, the one with minimum twist. It is basically a conversion activity.

So if design is defined as a search effort, flattening cannot be design since no optimization or search is required. This holds when the rules governing flattening are
Figure 5.36: Two 3D wiring harnesses and their flattened representations, generated by the *f3flat* application.

Figure 5.37: Screenshot of the *f3flat* graphical user interface
5.7. Discussion: is this design correct, i.e. they match the wiring harness characteristics in the real-world. In case of the CAD systems in use today, inconsistencies are introduced in the flat model with respect to 3D, which triggers iterative improvements of that result, giving an illusion of design.

Chapter 8 will show that the time required to analyze and flatten a batch of wiring harness models from a commercial aircraft project is reduced to minutes using the f3flat KBE application. The following chapter will focus on the next hurdle in creating a formboard: fitting the wiring harness to a frame.
Chapter Six

Fitting and configuring a flat wiring harness for manufacturing

The automation of formboard design process steps concerned with the 3D domain were shown in chapter 5. As was the case for the analysis and flattening process steps, current methods for fitting and dress-up are time-consuming and repetitive. In this chapter automation strategies for fitting and configuring are selected and implemented in a Knowledge Based Engineering (KBE) application.

Fitting a wiring harness within a formboard frame is a search problem with multiple objectives. Section 6.1 presents these objectives, explains the current approach for fitting and discusses the main automation challenges. In order to reduce the search space a parameterization approach is proposed in section 6.2 that divides bundles in bendable sections. To ensure these bend sections meet physical constraints, the wiring harness 3D model is revisited to obtain bending limits for the flattened model.

Different approaches to automatic fitting are experimented with as described in section 6.3. A direct method based on manual fitting heuristics provides an effective way to obtain formboard layouts suitable for manufacturing. This is followed by the options for the addition of manufacturing instructions to a formboard configuration in section 6.4, also considering manufacturing ergonomics. The KBE implementation of the final demonstrator application f3fit is presented in section 6.5. The chapter concludes with some remarks on limitations and opportunities of the developed approaches in section 6.6.

6.1 A multi-objective search problem

The flattened wiring harnesses illustrated in figure 5.36 are spread out and some components overlap. The flattened harness is impractical to manufacture and its bundles must be rearranged to fit an acceptable formboard table frame. The main objectives of fitting a wiring harness configuration within a formboard frame are:
Minimize table size  A smaller table size reduces the distance manufacturing operators must reach and walk. Another reason to aim for small tables is that it reduces the shop floor space required per wiring harness. Given standard table frame sizes the objective is to fit the wiring harness within the smallest frame.

Eliminate component crossings and overlaps  Component crossings and overlaps are impractical for manufacturing and limit the quality control function of the formboard drawing. In principle all crossings should be eliminated, but this may not be possible in all cases.

Minimize the number of bends  Wire bundles are kept in place on the formboard by means of pins or nails. More pins are required for bends than straight sections: more bends means more time is required to prepare the formboard table. The number of bends introduced for fitting needs to be minimized, this excludes flattened bends that must be respected with respect to the 3D geometry. Another reason to aim for straight sections is to minimize the length differences between wires in a bundle. Also, bends introduced on the formboard drawing must be checked for flexibility limit violations.

Optimize for manufacturing ergonomics  The formboard configuration affects the work posture of a manufacturing operator. An operator’s reaching distance should be taken into account when placing components on the formboard and selecting a formboard size. Components that take more time to assemble (e.g. large connectors) should better be placed at more ergonomically favorable positions than components requiring less effort. Manufacturing ergonomics are further influenced by the spacing between components, the clarity and visibility of symbols, uncluttered layout (i.e. straight, aligned sections). Where for the first three objectives a clear numerical criterion can be defined (i.e. table width, height, number of overlap points, bend angles and radii), this is not so straightforward for manufacturing ergonomics. Section 6.4.1 discusses an approach to evaluate the ergonomic performance of a fitted formboard configuration.

The fitting problem is in fact a multi-objective search problem. Note that a balance between certain (contradictory) objectives must be found. Limiting the number of bends for example may require a larger frame size. Figure 6.1 typical Degree(s) Of Freedom (DOF) related to fitting a wiring harness on a formboard. The many different ways the wiring harness could be manipulated to find an acceptable fitted configuration make this a search problem. It is a Multidisciplinary Design Optimization (MDO) problem as the objectives concern different disciplines such
as mechanics and manufacturing ergonomics that cannot be handled by simple lower and upper limits as the design variable values.

Figure 6.1: Schematic illustration of the fitting problem and its degrees of freedom.

Figure 6.1 shows both local and global degrees of freedom. Each bundle section can be bent at one or more positions with a certain angle and bend radius. They can also be flipped, which means the orientation of the next break-out is reversed. At each break-out point, the direction of bundles could be modified as well. The position and orientation of the entire wiring harness are two global degrees of freedom.

The harness configuration is also subject to a number of constraints. The main constraint is of course the wiring harness model itself: its structural configuration and dimensions must be respected. Especially the bundle length is an important constraint, its length must be the same as in the 3D design. Other important constraints are component orientation and flexibility limits of bundle sections in terms of bending and twisting.
6. FITTING AND CONFIGURING A FLAT WIRING HARNESS FOR MANUFACTURING

6.1.1 Current approach

This MDO problem is currently solved manually with the formboard engineer as the optimizer. The objectives are evaluated based on experience and consultation of handbooks while converging to a satisfactory final solution. For a Fokker Elmo formboard engineer, fitting is the most creative task in making a formboard drawing. CATIA V5 [14] is the environment in which the wiring harness is flattened and subsequently fitted. The engineer selects a formboard frame size estimated to be of sufficient size and imports the flattened model. Some issues are presently caused by the CAD system interface itself such as only allowing a user to work on a single zone at a time (harnesses can consist of many zones, see section 2.1) while hiding other zones. Such inconveniences make the process more time-consuming than necessary, but might be less of a problem in CATIA V6 [23].

The main manipulation activities are manually defining bends (i.e. ‘rolling’ a bundle) and modifying break-out directions. The latter is also used to perform bundle section flipping. In order to ensure physical constraints (bending, twisting limits) are met, the fitted result is compared to the 3D model and checked for potential violations using handbook methods.

The fitting objectives are typically evaluated qualitatively and results vary per formboard engineer and aircraft program. Flexibility constraints are checked only when deemed necessary, depending on the experience of the formboard engineer. Manufacturing ergonomics are accounted for based on the personal judgment of the formboard engineer, who may get feedback from operators in some cases. Fortunately a number of formboard engineers have manufacturing experience themselves, and are actually good at creating ergonomically favorable drawings.

6.1.2 Automation challenges

Objective formulation The first challenge in automating the fitting process is formalizing the objectives for a formboard layout. In the existing process a layout is evaluated qualitatively based on general guidelines, reference drawings and especially (co-worker) experience. Formboard drawings created with a (semi)automated fitting process should be ‘as good as’ or ‘better than’ manually designed formboard drawings, when assessed by a formboard expert. Formal pass/fail criteria and solving strategies must be devised for the objectives that match such qualitative assessment. Weighing and prioritizing among the (contradictory) multi-disciplinary objectives will be required, and may need to be variable, as the importance of objectives typically varies per aircraft program and its development phase maturity.
6.1. A multi-objective search problem

**Size of the solution space** Fitting is a multi-objective search problem with a practically infinite solution space. This is because variables such as the position of a bend and the bend angle are continuous, i.e. any position or angle could be selected. Formboard engineers do this based on their experience and their spatial visualization ability, which results in a wide variety in bend radii, etc. An appropriate parameterization or discretization of the wiring harness for fitting is required. The parameterization approach should limit the solution space sufficiently to allow finding a feasible solution in reasonable time while providing a sufficiently large design space to obtain a feasible formboard drawing. Section 6.2 details the parameterization approach.

**Search algorithm** Unfortunately no directly comparable optimization/search problems were found in literature. A technique with perhaps the most similarities to the problem at hand is Graph Drawing. This field concerns the geometric representation of graphs and networks, mainly aimed at information visualization [120] However typical graph drawing algorithms do not maintain fixed lengths between nodes, do not account for flexibility constraints and are not aimed at satisfying the specific (multi-disciplinary) objectives associated with formboard design. The main challenge will be to develop an effective search algorithm. As will be seen in section 6.3, both domain-specific heuristic and meta heuristic search methods are experimented with to solve the fitting problem.

**Structural configuration variation** The search method should be sufficiently specific to obtain formboard configurations meeting the search objectives while being sufficiently generic to do this for the majority of wiring harnesses. No two wiring harnesses are identical, while the basic components are the same, harness topologies or structural configurations are different. Every wiring harness will have a different design space. Developing a method suitable to fit the majority of wiring harnesses is a greater challenge than developing one aimed at solving a single wiring harness.

The fitting process has two main time-consuming elements: manipulating the harness model and performing checks on the objectives and constraints. Automating the former requires the use of search techniques and approaches for this are detailed in section 6.3. But even without automating the fitting task itself, time can be saved by eliminating the need to perform extensive checks. In practice this means a user will either be automatically constrained to operate within allowable limits or be given warnings in case of constraint/objective violations, as is presented in the next section.
6.2 Parameterization approach

The DOF introduced in figure 6.1 need to be translated in a parameterization approach for the wiring harness fitting High Level Primitive (HLP)s. The approach should provide a basis for both manual and automated fitting. Automated fitting features geometry-evaluating functions that can be computationally expensive. Manual fitting should still be possible to allow for variations in wiring harness configurations. In this way it will be possible to create formboard drawings in case automated results are not entirely satisfactory. These use cases translate into two objectives for the parameterization approach:

- Restrict the solution space: limit the number of configurations that can be generated (i.e. restrict the range of variables) to reduce computation time.
- Constrain a user or search algorithm to operations within allowable limits.

**Position & orientation** The position and orientation of the flat wiring harness become relevant only when a formboard frame has been selected. Until then no boundaries exist, a position and orientation are only needed to visualize the model. Once a frame is selected the wiring harness can be positioned within its borders by translating the reference connection point \( C_{ref} \) along a vector \( V_{tr} \) (where \( V_{tr} = f(L_{frame}, H_{frame}) \)) to its new position \( C'_{ref} \), as illustrated in figure 6.2. The orientation of the wiring harness is defined by the reference tangent vector \( T_{ref} \) with respect to the same reference connection point \( C_{ref} \), see figure 6.3.

![Figure 6.2: Schematic illustration of the wiring harness position DOF, defined by the variable \( C_{ref} \) of the wiring harness.](image)

**Break-out directions** Break-out directions \( B_i \) (with \( i = 1, 2, \ldots, n \) for \( n \) bundles) at a connection point \( C \) are often modified to fit a wiring harness. The break-out angles
6.2. Parameterization approach

Figure 6.3: Schematic illustration of the wiring harness orientation DOF, defined by the variable $T_{ref}$ of the wiring harness.

$\beta_i$ of each connection point can be changed by applying a rotation angle $\Delta \beta_i$ as depicted in figure 6.4. In generating the flattened wiring harness these directions have already been changed by rounding the angles to a certain value and imposing a minimum break-out angle between bundles, as explained in section 5.5.2. No other rules are available however to limit further deviations from the direction most closely matching the 3D connection point. As long as no strict rules are available, modifying break-out angles should not be automated to maintain the orientation with respect to the 3D model. Therefore this should only be done manually by an experienced formboard engineer.

Figure 6.4: Schematic illustration of the break-out direction DOF, defined by the variable $\Delta \beta_i$ of a connection point.

**Flipping** Reversing the orientation of a connection point can be done by twisting the connecting bundle by 180 degrees. This is flipping a bundle section. Whether it is allowed to do so is known from the twisting analysis performed in the 3D analysis phase (see section 5.4.2). When allowed, flipping of a bundle HLP can be controlled by means of the logical variable $flip?$, which is either true or false (i.e. $T$ or $F$).
Figure 6.5 shows a bundle between two connection points that is flipped, resulting in a reversal of the plane vector $\mathbf{B}_2$.

![Diagram showing flipping DOF](image)

Figure 6.5: Schematic illustration of the flipping DOF, defined by the logical variable $\text{flip?}$ of a bundle.

**Bending** The most essential wiring harness manipulation method for fitting is bending bundles. Figure 6.6 illustrates a bundle with a bend section with two variables: a bend angle $\theta_{\text{bend}}$ and bend radius $R$. A zero degree angle gives a straight section, while a non-zero angle results in a bent section. Just as bending flexibility is an issue when straightening a bundle (see section 5.5.2), it can be an issue when modifying its curvature for fitting. The bending flexibility could be evaluated after or before bending a section. In practice, evaluating whether a certain bend violates flexibility limits is computationally relatively expensive.

![Diagram showing bending DOF](image)

Figure 6.6: Schematic illustration of the bending DOF, defined by the bend angle $\theta_{\text{bend}}$ and the bend radius $R$ of a bend section.

The flattening and fitting steps are implemented in separate stages as was explained in section 4.4. There are two options to evaluate whether flexibility violations have occurred:

1. Evaluate after bending.
2. Evaluate before bending.
6.2. Parameterization approach

- **After fitting** - A function in the analysis & flattening application *f3flat* could be called to compare the fitted geometry state to the 3D geometry state. This function would indicate whether the fitted geometry violates flexibility limits or not.

- **Before fitting** - Bundles can be discretized (in the *f3flat* application) into bendable segments with predefined bend radii which can be analyzed for violations of flexibility limits. The input for the *f3fit* fitting application will then consist of predefined bendable segments, each constrained by a fixed bend radius and maximum bend angles.

The first option does not restrict the solution space nor does it provide upfront constraints to a user. It may be needed for fitting strategies requiring more flexibility in the design space however. The second option constrains the model to a discrete number of formboard configurations and eliminates the need for computationally expensive flexibility checks during fitting steps. The main drawback is that it can eliminate good configurations from the design space. The discretization of a bundle into bend segments is selected as parameterization approach for bending, more details are given in section 6.2.1 and the approach to evaluate flexibility limits is given in section 6.2.2.

With the fitting model parameterized with predefined bends, a user can safely fit an entire wiring harness without having to worry about violating physical constraints. Warnings can be given to indicate crossings with other components or the selected formboard frame. The parameterized fitting model with predefined bends is illustrated in figure 6.7. Note that areas are reserved to accommodate dress-up symbols; overlaps of these areas will trigger warnings as well. Fitting a harness manually with this approach can reduce the time needed for the fitting process by eliminating the need to perform manual checks.

### 6.2.1 Bend assignment

The parameterization of bends in a bundle section can be made dependent on its flexibility parameters, including the bundle diameter. The first few centimeters of a bundle from a break-out point or an endpoint connection are less flexible than the rest of the bundle and must therefore remain straight. This is why in the parameterization, a bend section cannot start at the connection point but only after a minimum distance $d_{cpt,min}$. This is illustrated in figure 6.8. The value for $d_{cpt,min}$ can be selected from a lookup table based on the bundle diameter $D_{bundle}$ and flexibility category (which depends for example on the covering materials). Subtracting
Figure 6.7: The wiring harness model for fitting. Screenshots from the f3fit application.

Figure 6.8: Schematic illustration of the bend assignment parameters.
6.2. Parameterization approach

these distances from the bundle length leaves the total length available for bends $L_{available}$.

The bendable length $L_{available}$ is divided in alternating bendable and straight sections. The length a bend section $L_{bend,max}$ depends on the standard radius $R_{bend}$ and the target maximum bend angle $\theta_{target}$. The former can be selected from the same lookup table as $d_{cpt,min}$. The $\theta_{target}$ can be selected by the user, as larger or smaller angles may be preferable, depending on fitting approach. These bends of standard length are distributed along the bundle. The length of a straight section between bends $d_{bend}$ depends on the distribution of bend sections along the bundle. An equispaced distribution as illustrated in figure 6.8 can be used (with $d_{bend} = \frac{L_{available} - nL_{bend,max}}{n-1}$ for $n$ bend sections), but other distributions could be implemented as well.

The use of standard bends sizes has the potential added benefit of more uniformity in formboard layouts and may provide an opportunity to standardize production tooling. A possibility for the latter is to replace the many nails/pins required for a bent section by a single mold, thereby reducing formboard set-up time on the shop floor.

6.2.2 Determining bending limits

The target angle $\theta_{target}$ (see figure 6.8) is the maximum angle a bend section is intended to bend over. It remains to be checked whether a bend with a certain angle and radius falls within flexibility constraints. To this end the bending analysis as detailed in section 5.4.2 is applied again. In the flattening analysis case the evaluation was based on the difference between the curvature in the 3D state and the flattened state (i.e. no curvature at all, or straight), see left side of figure 6.9. For fitting the curvature of a bent section must be compared to the curvature of the section in the 3D state. Figure 6.9 shows that three cases exist. It shows an example section with a constant radius of curvature in its 3D state. In case:

1. The bend brings the fitted state closer to the 3D state compared to the straightened (i.e. flattened) state, so no evaluation is required as: if $R_{3D,in} < R_{bend(1)}$, then $R_{min} < 1/\sqrt{\kappa_{3D,out}^2 + \kappa_{bend(1)}^2}$ holds, where $\kappa_{bend(1)} = 1/R_{bend(1)}$.

2. The fitted state is a bend in the opposite direction with respect to the 3D state. As $R_{min}$ is defined with respect to a straight section, the evaluation of the bend radius is performed as if the bend was straight and adding the
bend curvature to the 3D curvature. So the bend is allowed if: \( R_{\text{min}} < 1/\sqrt{\kappa_{3D,\text{out}}^2 + (\kappa_{\text{bend}(2)} + \kappa_{3D,\text{in}})^2} \), where \( \kappa_{\text{bend}(2)} = 1/R_{\text{bend}(2)} \).

3. The fitted state is a bend in the same direction as the 3D state, but bent further. In this case the fitted bend radius is compared to the straight section:
\[
R_{\text{min}} < 1/\sqrt{\kappa_{3D,\text{out}}^2 + \kappa_{\text{bend}(3)}^2}, \text{ where } \kappa_{\text{bend}(3)} = 1/R_{\text{bend}(3)}.
\]

As the curvature of the 3D state is not constant in practice, the actual implementation applies this approach to the curvature distribution. Note that the total radius of curvature must be compared to the limit value \( R_{\text{min}} \), which is why the out-of-plane curvature component \( \kappa_{3D,\text{out}} \) is included. The rule for case 3 is conservative, if a straight section can be bent with \( R_{\text{bend}(3)} \), a section with \( R > R_{\text{bend}(3)} \) can certainly be bent to \( R_{\text{bend}(3)} \). The rule for case 2 assumes that it is easier to bend a section from \( \kappa_a \) to \( \kappa_b \) (where \( \kappa_a \) to \( \kappa_b \) have opposite signs) than from \( \kappa = 0 \) to \( (\kappa_a + \kappa_b) \). This rule is also estimated to be on the conservative side by Fokker manufacturing engineers. Experiments testing the flexibility of bent bundle sections could be performed to improve these bending analysis rules.

For every bend section, the flexibility is evaluated for the case the section is bent by the angle \( \theta_{\text{bend}} = \theta_{\text{target}} \) over the standard radius \( R_{\text{std}} \), both in Clockwise (CW) and Counterclockwise (CCW) direction. If the minimum bend radius \( R_{\text{min}} \) is violated, the bend angle \( \theta_{\text{bend}} \) is decreased a step and a new evaluation is performed. This procedure is repeated until an allowed maximum angle is found (i.e. \( \theta_{\text{max},\text{CW}} \) and \( \theta_{\text{max},\text{CCW}} \)). The limit value of \( \theta_{\text{max},\text{CW}} \) and \( \theta_{\text{max},\text{CCW}} \) is zero, as the straight state is allowed by definition.

Figure 6.9: Evaluation of the bending limit is required when the bend falls outside the range set by the 3D and the straight bundle geometries.
6.3 Fitting approaches

Flipping a bundle section changes the orientation of the in-plane curvature along the bundle. Therefore the maximum bend angles must be calculated for CCW and CW bends in both the flipped and the not-flipped case. With these four limits \((\theta_{\text{max,CW}}, \theta_{\text{max,CCW}}, \theta_{\text{flipped,CW}}, \theta_{\text{flipped,CCW}})\), the bends are constrained such that only allowable configurations can be created during fitting. The next section discusses automated fitting approaches.

6.3 Fitting approaches

Although it is already somewhat restricted, the parameterized model offers a large range of wiring harness configurations. This is a large combinatorial problem: a harness with \(n\) bends where each bend is given \(k\) angle options gives \(k^n\) combinations. The number of combinations increases further when the the flipping options are included: \(2^m\) (where \(m\) is the number of bundle sections that can be flipped).

In practice the number of bends will vary between a dozen for a small harness to hundreds for larger harnesses. Allowing bending from 45 degrees CW to 45 degrees CCW with 15 degree steps results in 7 bend angle options (e.g. 25 bend positions: \(7^{25}\) combinations).

This section discusses the application of different search techniques to obtain reasonable fitted formboard configurations from this vast solution space. For these techniques the path to a satisfactory configuration is not important, only the final state is. Different approaches have been experimented with (section 6.3.1) and a practical approach was selected and developed (section 6.3.2).

6.3.1 Experimentation

The first approach aims at resolving crossings in the flattened model in a straightforward manner. A global function identifies every crossing point and the bundles corresponding to each point. For each bundle bends are identified as well as the tangent vector at the crossing. The tangent vectors provide the angle between the two bundles. A set of case-based rules determine and apply bend angles to diverge the bundles. An example is a rule to determine at which distance from the crossing a bend should be applied, depending on bend availability and existing bending. The model is modified based on the computed bend angles. The changed configuration can feature (other) crossings, triggering a new iteration. This simple approach was soon found to be infeasible. It was not able to eliminate all bundle crossings, even with very simple test cases. Tweaking of the bend angle application rules would be required for every harness configuration.
Another approach treats each bend as an agent that iteratively changes its angle $\theta_{\text{bend}}$ based on the evaluation of a penalty function $f_P$. The penalty function increases when the branches connected to the bend come closer to other components (i.e. it decreases towards empty space). As straight sections are favored the penalty increases with the bend angle magnitude. Crossings trigger a penalty in bends located before a crossing point, and the penalty increases closer to the crossing point. Each component of the penalty function ($f_P = f_{P,\text{cross}} + f_{P,\text{bend}} + f_{P,\text{comp}}$) is illustrated schematically in figure 6.11. The formboard frame is introduced as an upper and lower bound acting as any other harness components with respect to the penalty function evaluation. The concept was to start with a large frame and reduce it during the iterations, effectively 'squeezing' the harness within the frame.

An iteration step starts with a wiring harness configuration in which each bend evaluates the penalty function ($f_P(\theta_{\text{bend}})$). The penalty function is evaluated again for the state of the wiring harness corresponding to neighboring bend angles in CW and CCW direction (i.e. $f_P(\theta_{\text{bend}} + \Delta \theta)$ and $f_P(\theta_{\text{bend}} - \Delta \theta)$). A greedy selection method is used: the state with the lowest penalty is selected for the next iteration. Simultaneous (multiple bends modified per step) and sequential iteration strate-
gies (one bend modified per step) were experimented with.

\[ \theta_{\text{bend}} = \theta \quad \theta_{\text{bend}} = \theta + \Delta \theta \quad \theta_{\text{bend}} = \theta - \Delta \theta \]

Penalty function component
\[ f_{P, \text{bend}}(|\theta_{\text{bend}}|) \]
The bend angle magnitude.

\[ f_{P, \text{cross}}(d_{\text{crossing}}) \]
The distance to a crossing point.

\[ f_{P, \text{comp}}(d_{\text{comp}}) \]
The distance to other components.

This approach gave reasonable results for a few simple test cases, but in most cases bundle crossings were not eliminated. Using small bend angle step sizes proved more stable running, but was also more time-consuming. The results were often locked into an unfeasible configuration, mainly due to the presence of many successive crossing points in different bundles. An approach to eliminate all bundle crossings before proceeding to further fitting steps was developed, inspired by rolling up a fire hose: the so-called roll-up, roll-off approach.

**Roll-up, roll-off approach**

The basic concept of the roll-up roll-off approach is to assign maximum CW bend angles to a set of succeeding bend sections, i.e. 'roll them up' as illustrated in figure 6.12. Step by step each successive bend section is given a CCW deflection, while checking for crossings with the main branch and the already fitted components. This 'rolls off' the bundle sections tightly to the other geometry without crossing it.

The method identifies the main branch which splits the harness in two sides that are solved separately. The main branch bundles are not bent, see figure 6.13. For
the upper part of the harness the first endpoint at the left side that is not part of the lower part of the harness is selected as a starting point (A in figure 6.13). Every next endpoint is selected by traversing the harness depth-first (i.e. follow the connecting bundle in clockwise direction until encountering an endpoint). Note that in the example all bundles are rolled-off to the left, but different patterns could be used as well.

Doing this for every set of bends between two endpoints gives a tightly fitted
configuration without any crossings/overlaps between components. The configuration would not qualify as a suitable layout for manufacturing due to the many bends, it can however be used for the following:

- As a starting point for other search methods to find a manufacturable configuration.
- As a technique to provide a clear, structured starting point for manual fitting operations.
- To define an initial minimum size for the formboard frame.

The method was used to obtain an initial configuration in a set-up using a genetic algorithm to fit the wiring harness. Another example is given in figure 6.14, note it includes bends to be respected.

**Genetic algorithm** An optimization using a genetic algorithm (see section 3.1.2) was developed to fit a wiring harness to a formboard [121]. Genetic algorithms are a type of meta heuristic search method, allowing an optimization algorithm to escape from a local optimum. The objectives of the optimization were to:

1. Maximize the free space per bundle
2. Minimize the number of bends and the bend angle magnitudes
3. Remove crossing violations

The fitness of a bundle with respect to these objectives is computed as follows: (1) The free space is the sum of the minimum distances from bundle sections to other bundles in the model. (2) A reward is given for a straight bend section and penalties are introduced for bends increasing with the bend angle. (3) Bundle sections following a crossing point receive a fixed penalty (i.e. the penalty increases when a larger part of a bundle comes after a crossing point). The multi-objective optimization problem is transformed to a single objective by weighing the objectives with respect to each other. The fitness function is the sum of the weighed fitness evaluations of all bundles.

---

1 The genetic algorithm was developed during the Master thesis work of Maarten Nelissen. The objectives went beyond the implementation of a genetic algorithm and include optimization to limit configuration vulnerability to changes. To this end a change simulator and a refitting algorithm were developed as well [121]
Figure 6.14: Results of applying the roll-up, roll-off method to a test harness. Image generated by the f3fit application.

The optimization was implemented by adapting an existing open source genetic algorithm engine [122]. The procedure followed is illustrated in figure 6.15. One candidate of the initial population is the tight-fitted harness using the roll-up roll-off method. To improve this candidate a straightening algorithm removes unnecessary bends. Other candidates are each generated by randomly creating 20 candidates and selecting the fittest one not already in the population. Iterations start that will either terminate after a fixed number of generations or when a fitness threshold is reached.

Candidates are selected from the population for reproduction in order to obtain a new generation by tournament selection. During crossover parent bend angle lists are split and recombined to form child bend angle lists. Each bend angle may
6.3. Fitting approaches

Generate initial population $N=0$

Determine fitness values

fitness threshold?

OR $N_{\text{limit}}$?

Selection

Mutation

Crossover

New generation $N=N+1$

Elite children

Local search

yes

no

Figure 6.15: Flow chart of the genetic algorithm. Adapted from [121].

randomly be changed based on a mutation rate. Creating children by mutation allows jumping over local optima. The fittest candidate from the old generation may not be selected for the new generation due to the tournament selection approach. In those cases this candidate is added to the new generation as an 'elite' child. To further improve the new generation a local search is performed on a selection of candidates. This local search consists of creating close neighbors of a candidate by changing a single bend. Neighbors of these neighbors are created by subsequently selecting and modifying a bend close to the previously changed bend. Repeating this procedure results in a 'local-city' of neighbors of which the fittest are selected for the new generation.

A global optimization of the wiring harness is less likely to get stuck in a local optimum, but takes more time to generate a feasible solution than sequential local optimization of branches. However, the latter gives a higher risk of ending in a local optimum. A hybrid approach starting by sequentially optimizing branches and subsequently performing a global optimization proved more effective.

The geometrical evaluations were expensive in computation time leading to evaluation times in the order of hours [121] for configurations such as shown in figure 6.16. Computation times can be expected to further increase with larger harnesses. However it must be noted that no clear convergence criterion was set and the algorithm was often stopped only after reaching the predefined maximum number of generations. Some limitations of the implementation are that flipping, as well
as adjusting the position and orientation with respect to the formboard frame were
not accounted for and the objectives did not include horizontal alignment. Parallel
with the development of the genetic algorithm, an approach was developed based
on formboard engineering heuristics that aimed at obtaining configurations resem-
bling existing drawings while being faster and more easy to use alongside manual
fitting: the alignment method.

6.3.2 Alignment method

A fitting approach was developed inspired by the heuristics of manual fitting by
formboard engineers. Observations of formboard engineers fitting a wiring harness
using CATIA provided insight in typical fitting practices. A formboard engineer of-
ten focuses on fitting a part of a harness and adjusts the complete configuration in
later steps when a better overview is attained. This lead to adopting an approach
where the harness is fitted per branch. Typical manual fitting heuristics are for ex-
ample ‘move bundles breaking out to one side’ and ‘bend a bundle breaking out
outwards’. These heuristics were translated into fitting strategies to be applied to
individual branches. A branching wiring harness product model is adopted, as was
illustrated in the bottom-right of figure 4.14.

The main concept is to align the bundles of each branch as tightly as possible
6.3. Fitting approaches

Fit-tight the wiring harness by alignment Select and position formboard frame Adjust layout within frame using flipping and bending strategies for better use of free space and improved ergonomics

Figure 6.17: Basic steps of the alignment method.

(i.e. bend bundles to align with their parent branch without crossing). Then select the smallest formboard frame which the aligned configuration fits in and perform adjustments to better use free space and improve the layout from an ergonomics perspective. These steps are illustrated schematically in figure 6.17.

**Flipping strategies**  Consider a branch with 6 in-line bundles and 5 branching bundles such as shown in the left of figure 6.18. If each in-line bundle is physically allowed to flip there are $2^6$ possible combinations. The number of combinations is reduced by evaluating only flip combinations according to specified strategies. Figure 6.18 shows three flipping strategies: not flipping any bundle at all or 'none', 'per-direction' and 'all-on-open-side'. The 'per-direction' strategy aims at reversing break-out points such that bundles break-out in opposite directions on each side of the branch. The 'all-on-open-side' strategy aims at getting as many bundles breaking-out on the open side of the branch. This is the side where the angle between the branch and its parent is largest. It is often not possible to achieve the aims of these strategies because flipping is not allowed or because bundles break-out in multiple directions at a connection point. In such cases the algorithm attempts to achieve the strategy aims as well as possible.

Figure 6.18: Schematic illustration of flipping strategies.
Bending strategies Consider the branch shown at the left of figure 6.19, different strategies for aligning the branching bundles can be applied. Except for the 'none' strategy (i.e. no bending applied) each strategy aims at aligning the branching bundle with the parent branch in-line bundles. One approach is to use the smallest bend angle: the 'min-angle' strategy. Alternatively, the 'to-direction' or 'from-direction' strategies can be used, which aims for alignment towards or away from the branch connection point respectively. The sequence in which branching bundles are bent depends on the strategy used to prevent locked configurations. For the 'to-direction' strategy the branching bundle closest to the connection point will be aligned first, for the 'from-direction' it will be aligned last. Although figure 6.19 shows the bending strategies applied on both sides of the branch, the sides are independent and different strategies can be applied on each side.

![Bending strategies](image)

Figure 6.19: Schematic illustration of bending strategies.

More strategies can be thought of, but these provide a reasonable set for fitting as demonstrated in chapter 8. For one branch using the strategies of figures 6.18 and 6.19 $24 \ (n_{flip-str}(n_{bend-str-left} + n_{bend-str-right}) = 3(4 + 4))$ different configurations can be generated. To select the combination of strategies a fitness function is defined.

Tight-fit fitness evaluation The objective is to obtain a tightly fitted configuration without crossings. A penalty function $f_P$ is defined with two components. The area of the fitted branch bounding box $A_{branch}$ is used as a penalty value to minimize. Smaller values indicate a more tight solution. Violating the crossing constraint adds a penalty a few orders of magnitude larger than the area. Bend angles are not explicitly accounted for as larger bend angles also result in a larger area and unnecessary bends will be removed in later steps. As the bend strategy applied at one side of the branch is independent of the other side, the penalty function can be evaluated separately per side. The sides are defined as left and right with respect to the branch break-out direction. For a branch the results of the penalty function depends on the flipping strategy $F$ and bending strategies $B_l$ and $B_r$ (left and right) that are applied
6.3. Fitting approaches

to it:

\[ f_P(F, B_l, B_r) = A_l(F, B_l) + A_r(F, B_r) + A_0 k_{cross}(F, B_l, B_r) \]

where

- \( A_l = w_l h_l \) and \( A_r = w_r h_r \) (\( w \) is the dimension in the branch break-out direction and \( h \) in the perpendicular direction, see figure 6.20).
- \( A_0 \) is the total branch area for the case no strategies are applied.
- \( k_{cross} = 0 \) if no crossings are present or \( k_{cross} = 999 \) when a crossing is present on the left or right side of the branch.

An example case is given in figure 6.20, where the configuration with the minimum \( A_{branch} \) corresponds to the 'all-on-open-side' and 'min-angle' flipping and bending strategies, respectively. However, as this configuration has a crossing point, the penalty will be high. The combination of the 'none' and 'min-angle' strategies, indicated in figure 6.20, results in the smallest penalty.

Figure 6.20: Schematic example of applying the bending and flipping strategies to a branch. The objective is to minimize the branch area \( A_{branch} \), without crossing points.

**Search iterations** The wiring harness is fitted by iteratively searching for the best combination of flipping and bending strategies. It starts with the outer branches, proceeding inward until all branches are aligned with respect to their parent branch. For each branch an exhaustive search of all strategy combinations is performed and the strategy with the lowest penalty is selected (i.e. greedy selection).
The alignment steps must start at the lowest branch nodes because only the geometry of the branch itself and its subbranches is considered when evaluating the fitness of a branch. Consider the test harness of figure 6.21, where branches have been highlighted per level in the product tree. The first alignment is performed for the lowest branches (i.e. 1.1.1, 1.1.2, 1.1.3). These do not have branching branches, so no alignment operations are required as indicated in figure 6.22. Branch 1.1 is the first to be aligned, which is illustrated in the top-right of figure 6.22. The most favorable configuration uses the 'to-open-side' and 'min-angle' strategies. Traveling up in the branch tree more alignment evaluations are performed, resulting in the tight-fitted configuration shown in the bottom-right of figure 6.22. A more detailed description of the function implementation is described in section 6.5.2.

**Frame selection** The tight-fitted configuration provides a bounding-box for which the smallest standard formboard frame size can be selected. Section 6.4.1 discusses these standard frame sizes. The wiring harness model is automatically positioned in the center of the frame. As the center image in figure 6.23 shows, there is free space available and many bends seem to be unnecessary.

**Layout adjustments** Adjustment functions have been developed to improve the layout with respect to the tight fitted configuration. The main adjustment function is the 'relax' algorithm. This method iteratively attempts to shift bend angles from one position to a downstream position (i.e. towards and endpoint) and evaluates whether the change improves the layout.

This evaluation is done by means of a layout fitness function, defined with respect to the bend section (B) to be adjusted in the 'relax' algorithm iteration. This function is equal to the minimum distance of all sections succeeding section B (i.e.
1. No branching in 1.1.1, 1.1.2, 1.1.3
2. No branching in 1.2, 1.3, 4.1
   - Alignment of branch 1.1
3. No branching in 2, 3
   - Alignment of branch 1
   - Alignment of branch 4
4. Alignment of the main branch

Figure 6.22: Alignment steps for the 'MOD3' test harness configuration of figure 6.21

Figure 6.23: A formboard frame is set around the tight-fit result of the 'MOD3' test harness and adjusted using the 'relax' algorithm.
downstream sections) to all sections preceding section B (i.e. upstream sections) and the formboard frame. The objective in a 'relax' algorithm iteration step is to maximize the function:

\[ f_{\text{layout}} = d_{\text{min}} = \min(d_{1,3}, d_{1,4}, ..., d_{i,j}, ..., d_{n,m}, d_{1,\text{frame}}, d_{2,\text{frame}}, ..., d_{n,\text{frame}}) \]

where

- \( d_{i,j} \) is the minimum distance from section \( i \) to section \( j \).
- \( d_{i,\text{frame}} \) is the minimum distance from section \( i \) to the formboard frame.
- Sections \( i = 1, 2, ..., n \) are the \( n \) downstream sections.
- Sections \( j = 3, 4, ..., m \) are the \( m \) upstream sections, minus the first two sections preceding bend section B. These, and bend section B itself are ignored from the calculation as these would yield (nearly) the same minimum distance for any adjustment of the angle of bend section B.

The 'relax' algorithm iterations follow a top-down approach instead of the bottom-up approach used for the alignment method. The first iteration attempts to shift the bend in the top-level branch by one bend position, which is selected in case the fitness improves (i.e. a greedy selection approach). The next branch in the tree is selected breadth-first and its bend can be shifted by one position, and so forth. When all the bends in all branches have had the opportunity to shift by one position, the second iteration starts again at the top-level branch to shift a bend to a further position. This continues until there are no bend positions left that improve the fitness. The result is illustrated in figure 6.23. The implementation of the algorithm is discussed in section 6.5.2.

Other layout adjustment methods have been implemented such as a re-alignment method that shifts bends upstream (i.e. opposite of relax method). An endpoint flipping method was implemented as well which flips a bundle connecting an endpoint when that improves the layout according to the layout fitness function.

The alignment method can automatically be applied to an entire wiring harness as described above, but also to a selection of or individual branches.

A wiring harness can be fitted manually within the constraints set by the parameterization (section 6.2) by selecting specific bend angles, flipping, as well as by applying bending and flipping strategies. Combining automatic and manual fitting allows the solving of a wide variety of wiring harness configurations. Typical
computing times for the full-automatic alignment method are in the order of a few minutes, see chapter 8.

6.4 Formboard configuration

The formboard drawing serves both as a manufacturing tool and a quality control instrument. Both pose requirements on the layout of the wiring harness and symbols used on the drawing. Section 6.1 introduced optimization for manufacturing ergonomics as an objective for fitting. The physical ergonomics of wiring harness manufacturing can be related to the formboard layout as described in section 6.4.1. The placement of manufacturing symbols on the formboard is one of the most repetitive and time-consuming tasks of a formboard engineer. It is also based on simple rules that can be captured and implemented in a KBE application as is described in section 6.4.2.

6.4.1 Manufacturing ergonomics

A study into wiring harness manufacturing ergonomics [39] shows manufacturing operators often work in ergonomically unfavorable postures. This is influenced by table and seat dimensions and the layout of components on the formboard drawing. For example, the average operator has to bend her back in order to reach components in the center of a table. Other industries than aircraft wiring manufacturing show that improving ergonomics decreases physical loads on operators, increases productivity and product quality [39].

Standard table widths of 61cm and 122cm are defined based on anthropometric data of females [123] (wiring harness manufacturing operators are typically women). A table width of 61cm allows comfortable reaching over that distance, although not for a long time [124]. Postures with vertical back and upper arm are optimal, therefore the optimal horizontal reach is about equal to the length of the lower arms, which is about 33 cm for Dutch women [39]. The vertical reach is important when manufacturing using tilted formboard tables (see figure 1.2). Its lower and upper limits are 76cm and 137cm for the average Dutch female worker aged 20-60 years [123]. The most favorable table height depends on whether work is performed seated (78cm) or standing (101cm) [39]².

²The research into wiring harness manufacturing ergonomics was performed during the Master thesis work of Joep van Heugten. The work included the design of an adjustable manufacturing table, of which a number were built for use at Fokker Elmo [39].
6. FITTING AND CONFIGURING A FLAT WIRING HARNESS FOR MANUFACTURING

To define an ergonomic component for the layout adjustment fitness function, a scoring system for formboard layout is developed. An ergonomic score is computed for each of the standard components of a wiring harness (bundle, endpoint, covering). The more work must be performed in less favorable postures, the higher the score a component will get. The score of a component consists of a positional score and a relative score. The former is illustrated in figure 6.24 and the second depends on the burden resulting from the presence of components and their positioning pins in the working area of a component. To allow for quick evaluation, the relative score is assigned per category as given in figure 6.25.

<table>
<thead>
<tr>
<th>Category</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>No obstructions</td>
<td>0</td>
</tr>
<tr>
<td>Single bundle</td>
<td>1</td>
</tr>
<tr>
<td>Single endpoint</td>
<td>2</td>
</tr>
<tr>
<td>Break-out or large connector</td>
<td>3</td>
</tr>
<tr>
<td>Multiple break-outs or endpoints</td>
<td>4</td>
</tr>
<tr>
<td>Multiple large connectors</td>
<td>5</td>
</tr>
</tbody>
</table>

Figure 6.24: Penalty distribution along the formboard width, for horizontal and tilted manufacturing. From [39].

Figure 6.25: Example of relative scoring. Adapted from [39].

The harmfulness of a working posture depends on the time that the posture is maintained. The position and relative scores are therefore multiplied by the time factor corresponding to a component. These time factors are based on norm times for manufacturing tasks [39]. An additional penalty is given to a component score
for certain special conditions, to name a few: bundles that must be twisted, because of the extra effort this requires; connectors that do not face the operator, making them harder to assemble; and components requiring special tooling (e.g. a heat gun). These rules are more specifically described in [39]. The complete scoring method may require some fine-tuning based on further testing. At date, only some of the scoring system elements are used in the layout adjustment fitness functions, in particular for the freeing of space around components and modifying endpoint orientations (see section 6.3.2).

### 6.4.2 Formboard dress-up symbols

Dress-up is the addition of manufacturing information to the basic wiring harness geometry. These additions can be figures corresponding to wiring harness components, for example to clarify geometric orientations, dimensions, identification codes, tolerance fields, destination codes, taping, twisting instructions, etc. Other information concerning the formboard is included as well such as tables with harness specifications, change notes, symbol legends and dimension lines.

These symbols, texts and figures are mainly manually added to the drawing, making it one of the more time-consuming aspects of formboard drawing design. The selection, sizing and positioning of most symbols follows simple rules, figure 6.26 shows a few that are typical for an endpoint component. The cutting and tolerance field symbols are positioned in-line with the bundle as it enters the backshell at distances which are a function of the endpoint dimensions. The clocking symbol faces the connector and its shape depends on the connector type and presence of a keyway angle. The creation of these symbols is relatively straightforward using a KBE system. This is one of the few steps in the development process that is already partially automated by custom CATIA scripts by Fokker Elmo.

The dress-up feature with perhaps the most complicated rule set was the routing number. Routing numbers are numbers that serve as way points for a manufacturing operator when routing individual wires on the formboard. The rules for assigning these to break-out points are: use 3 digits; start at the left side of the main branch with '010'; use numbers rounded up to tens for the main branch; numbers must increase while traveling from a break-out point towards an endpoint. These rules can be formalized into an algorithm as explained in section 6.5.3.

To ensure sufficient space is available for symbols on the fitted formboard, outlines of the most essential symbols are included in the crossing and fitness analyses discussed in section 6.3. The outlines of for example cutting and clocking symbols are treated as geometrical components that may not be crossed. These can be ob-
Figure 6.26: Examples of symbols related to a straight and an angled endpoint. Adapted from [39].

Figure 6.27: Different representation abstraction levels. The high abstraction was selected as the best option during dedicated focus group sessions with manufacturing operators. Adapted from [39].
served beside each connector in figures 6.16 and 6.23.

**Side note: information complexity** A lot of variation in symbols exist in the current aircraft and aero-engine programs at Fokker Elmo creating a sense of complexity and making it difficult to exchange manufacturing operators. A KBE application could in principle be set up to allow the full (confusing) variety in symbol styles. However, its development also provides an opportunity to reduce the mental load for operators by standardization and evaluation of which symbols actually need to be shown and hence standardized. Different philosophies with respect to manufacturing information representation exist: some favor a low abstraction level including all information on the drawing, while others favor an abstract drawing showing only basic geometry and provide information in separate documents (see figure 6.27). A set of information design requirements have been defined and sample symbols designed along these requirements have been proposed [39] and implemented in the KBE application.

### 6.5 KBE implementation

The parameterization and fitting approaches presented in the previous sections have been implemented in KBE applications. Figure 6.28 presents the general Unified Modeling Language (UML) activity diagram, starting with the 3D analysis and flattening as given in figure 5.23. The assignment of bend positions, evaluation of angle limits and definition export is implemented as the definition-collection Capability Module (CM) of the f3flat application, see figure 5.24. The fitting approach, addition of manufacturing information and export to a printable format are implemented in the f3fit application.

#### 6.5.1 Bend assignment

The definition-collection CM collects and stores the definition of the entire wiring harness HLPs in lists serving as input for the f3fit application. It uses the bundle-section-bends CM to define and analyze bends. These functionalities are implemented as CMs of the f3flat application because they require the 3D model and analysis results, but are not needed to perform the analysis and flattening steps.

**Assign bend positions** The bundle-section-bends class is mixed in the bundle-section class (see section 5.6) where the steps to analyze and flatten a bundle section are taken. As indicated in figure 6.29, bends are only assigned when a bundle section is straightenable. The program parameters provide a target bend angle
Figure 6.28: UML activity diagram of the $f3flat$ bend-assignment capability module and the $f3fit$ application.
\( \theta_{\text{target}} \). An external data file provides bend definition standards per flexibility category for specific bundle diameter ranges. With the bundle section flexibility category and its diameter, the corresponding standard bend radius \( R_{\text{std}} \) and minimum distance to connection point \( d_{\text{cpt,min}} \) can be determined. With the latter the bendable length \( L_{\text{available}} \) of the bundle section is computed from the bundle section length when connected to a connection point. These parameters determine the length of each bend \( L_{\text{bend,max}} \) and the maximum number of bends can be computed and positioned along the bundle section.

![Activity diagram of the assign bend positions activity](image)

Figure 6.29: Activity diagram of the assign bend positions activity of figure 6.28. See figure 6.8 for the geometry definitions.

**Evaluate deflection limits** Flipping a bundle section changes the in-plane component of the curvature. The deflection limits are therefore evaluated for both the regular and flipped cases as explained in section 6.2.2. Figure 6.30 shows that the in-plane and out-of-plane curvatures for the bundle section are computed by the bundle-section-bends class, which then instantiates the bundle-section-bend class for each bend position. This class incorporates a bend-angles-check function that is evaluated for both a positive and negative target bend angle (i.e. CW and CCW bends) and returns a bend angle limit.

**Export flat wiring harness definition** For each HLP a definition collection class is mixed in the HLP class. The definition-collection-bundle class uses the bundle-section-bends CM to determine the position and limits of bends. These
parameters and other attributes are collected and stored. The parameters needed for fitting are similarly collected for the endpoint and connection-point HLPs. The definition-collection-wiring harness class collects all HLP definitions and exports them to an output file file. This is the input file for the f3fit application. The file format and an example file are given in appendix I.

6.5.2 Fitting

The fitting approach as described in section 6.3.2 was implemented in the f3fit KBE application. The UML class diagram is given in figure 6.32 and shows the fitting-container-branches application Structuring and Collating Node (SCN) and its children. The alignment approach uses a branching product model structure, starting with the main-branch product SCN which is further detailed in figure 6.33. Figure 6.32 also shows a flat product model (wiring-harness product SCN), which contains connection-point, bundle and endpoint HLPs as direct children. This SCN was used for both the roll-up, roll-off method and the genetic algorithm experiments. The application SCN class further features a formboard-frame primitive, a table (used for 3D visualization) and the formboard-drawing CM to generate the complete formboard drawing for export.
6.5. KBE implementation

Figure 6.31: Activity diagram of the *export flat wiring harness definition* activity of figure 6.28

Figure 6.32: Class diagram of the *f3fit* application.
Generate f3fit harness model  The wiring harness definition is provided to the main-branch SCN in lists. A visualization of a main-branch instance was given in figure 6.7. A connection point $C_{ref}$ is selected as a starting point; this will typically be one in the center of the harness, connecting the thickest bundles (see figures 6.2 and 6.3). Figure 6.33 shows the class diagram, where the main-branch start-point and start-vector are provided to the connection-point point and reference-direction input-slots. All positioning and orienting is achieved similarly by means of a reference point and vector. Note that the diagram of figure 6.33 only shows a selection of class attributes, mainly some required for positioning and orientation. The bundle-directions list provides angles $\beta_i$ at which each bundle connected to the connection-point break-out (see figure 6.4). The connection-point class allows manual manipulation of these angles. The flipped? attribute indicates whether the connection point’s normal vector $P$ points into the positive or negative z-direction (see figure 6.5).

For each bundle breaking out of the main-branch connection-point $C_{ref}$ a branch is instantiated, which is a SCN that allows constructing the branching product model. The branch connected to the branch-connection-point is selected, all in-line bundles and connection points are identified and instantiated by means of the bundle and connection-point HLPs. In case the last connection-point connects an endpoint, an endpoint HLP is instantiated. For each in-line connection point, branches are generated per bundle breaking out.

A bundle consists of one or more sections which can be of three types: bundle-straight-section, bundle-fitting-bend-section (which can be given a bend angle for fitting) and bundle-fixed-bend-section (bends that could not be straightened). The start point and vector of a section corresponds to the end point and vector of the previous one. Bend parameters are provided by the bend-data list and the angles by the bend-data-angles list. Assigning a non-zero angle to a bundle-fitting-bend-section will create a bent section (e.g. figure 6.6), within the provided bend angle limits.

For a bundle segment that is allowed to be flipped a flipping class instance is generated. This class controls the flip status (flip?) of the bundle segment (which can cover multiple sections), based on flip settings from the flip?-list. A bundle-fitting-bend-section that is part of a flipped segment will automatically use the bend angle limits corresponding to the flipped case. The bend angle sign of a bundle-fitting-bend-section that is part of a segment succeeding a segment that is flipped will be reversed. Similarly, the flipped? attribute of a connection-point is switched after flipping the connecting bundle, which will re-
Figure 6.33: Class diagram of the main-branch product SCN of the f3fit application.
suit in mirroring the bundle-directions with respect to the connecting bundle centerline, as was illustrated in figure 6.5.

**Tight-fit harness by alignment method**  The alignment method is built up by a set of functions implemented in and called by the main-branch and branch SCN classes. To align the complete harness the *perform-fitting-iterations* function in main-branch is called, of which the activity diagram is given in figure 6.34. This function is recursive, it will call the *perform-fitting-iterations* function in its child-branches, which will do the same until arriving at the leaf branches. The alignment of a specific branch will happen only after aligning all its child-branches, thereby ensuring at bottom-up solving sequence.

![Activity diagrams of the perform-fitting-iterations and evaluate-branch-fitting-fitness functions, as implemented in the f3fit branch SCN.](image)

The alignment functions of the main-branch class are almost the same as for the branch class but for some simplifications, for example, no flipping needs to be
applied to the main-branch itself (as it has no bundles to flip about).

Figure 6.34 shows that after running perform-fitting-iterations for child-branches the perform-fitting-iterations function performs flipping and bending for all strategy combinations, while storing results and their fitness values. The fittest strategy combination is selected and the corresponding bend angles and flip-settings are applied to the branch.

The fitness is computed with the evaluate-branch-fitting-fitness function, given in the right side of figure 6.34. This function uses the crossing-test function to identify crossings. The crossing-test function basically tests the geometry of the child-branches of a branch with respect to each other and the branch bundles for possible coincident points. The evaluate-branch-fitting-fitness function also computes the branch bounding-box area (computed-slot of the branch class). These two parameters yield the penalty function \( f_P \) discussed in section 6.3.2.

The apply-flipping! function computes flip-settings for a given strategy. The input flip settings are used in case 'none' is selected, as indicated in figure 6.35. For the other strategies, the determine-flip-settings function is evaluated. This function receives as input the branch side (left or right) at which break-outs should be pointed from (from-side) or to (to-side) the branch connection point. In case of the 'per-direction' strategy, the from-side and the to-side should be opposite (e.g. from-side left and to-side right). For the 'all-on-open-side' strategy the open side of the branch is determined and both the from-side and to-side are set accordingly. The function returns a list of flip-settings for the branch which can apply these settings to the bundle instances.

For each branch side (with respect to the branch-connection-point) the apply-bending! function (shown on the left side of figure 6.36) can align child-branches to the parent branch. When more than one child-branch is present an alignment sequence is determined with the determine-alignment-sequence function to limit interference between branches. For the 'to-direction' strategy, the branch closest to the branch-connection-point is selected; for the 'from-direction' strategy, the branch furthest away from the branch-connection-point is selected; and for the 'min-angle' strategy, the branches are sorted by length, starting with the shortest ones. Each individual branch is then aligned with the determine-alignment-bend-angles function.

The determine-alignment-bend-angles function (right side of figure 6.36) computes the angle required for alignment and iteratively applies that angle to the bend sections of the branch, starting with the bend-section closest to the branch-
6. FITTING AND CONFIGURING A FLAT WIRING HARNESS FOR MANUFACTURING

Figure 6.35: Activity diagrams of the *apply-flipping!* function, as implemented in the f3fit branch SCN.

connection-point. The *proximity-test* function evaluates the distance between all sections before the bend section to all sections after the bend section. The iterations stop when an aligned configuration is where the distance is larger than a predefined minimum distance between components (e.g. 1mm). This will be the tightest fit of the branch.

**Set formboard frame** The formboard frame is a child of the fitting-container-branches SCN, see figure 6.32. This class contains the *set-frame-min-dimensions* function, which evaluates the minimum dimensions of the frame based on the tight-fitted harness configuration and assigns these to the formboard-frame *min-width* and *min-height* slots. The formboard-frame HLP subsequently selects the frame width and height from a list of standard dimensions. Another function in fitting-container-branches, *reposition-harness*, resets the *start-point* of the main-branch product model such that the harness geometry is positioned in the center of the frame (as in figure 6.2).

**Adjust harness layout** Several functions have been implemented to improve the wiring harness layout within the formboard frame. These functions are also implemented in the main-branch and branch SCNs use the layout fitness function $f_{layout}$ introduced in section 6.3.2. The simple activity diagram of the *evaluate-
6.5. KBE implementation

Figure 6.36: Activity diagrams of the apply-bending! and determine-alignment-bend-angles functions, as implemented in the f3fit branch SCN.

branch-fitness function that evaluates it is shown in figure 6.37. This function runs the proximity-test function discussed before with respect to the bent section in the branch. Additionally the minimum distance to the formboard frame is evaluated. The minimum value of these distances is used as a fitness measure to maximize.

To increase the available space around connectors for better manufacturing ergonomics, the evaluate-and-flip-endpoint function checks if flipping the connecting bundle increases the branch fitness value. If flipping is allowed and the fitness increases, the flipped configuration is kept as shown in figure 6.38. The evaluate-and-flip-endpoint functions are called by the main-branch function evaluate-and-flip-endpoints in a breadth-first sequence. This ensures endpoints connected to the principal (and often thickest) branches are flipped first.

To increase the space between branches, the main-branch relax-harness function is evaluated (figure 6.39). This function generates a breadth-first sequence (i.e. starting at the top, ending at the leaves) of all branches and iteratively calls the shift-a-bend function in each branch in order to trigger a downstream bend position shift. The shift-a-bend function first identifies the bent section(s), if present, and evaluates the layout fitness with respect to the bend section. The bend is shifted
6. Fitting and Configuring a Flat Wiring Harness for Manufacturing

Figure 6.37: Activity diagram of the *evaluate-branch-fitness* function, as implemented in the *f3fit branch SCN*.

Figure 6.38: Activity diagrams of the *evaluate-and-flip-endpoints* and *evaluate-and-flip-endpoint* functions, as implemented in the *f3fit main-branch and branch SCNs*, respectively.
downstream (i.e. towards endpoints) and the fitness is re-evaluated. For example, if bend sections 3, 4 and 5 would have had bend angles of 45, 30 and 0 degrees, the downstream shift would result in angles 0, 45 and 30 for these bend sections. The fittest configuration is kept and the procedure is repeated for the next branch. During the iterations, branches without bends or without room for improvement are pruned.

The re-align-branches function is similar to the relax-harness function, except that for each branch bend position both an upstream and a downstream shift are tested (using the shift-a-bend function), evaluated and the best option is selected. This two-directional search allows undoing bend shifts made by the one-directional relax-harness function for cases where that would improve the layout fitness.

### 6.5.3 Dress-up

The implementation of dress-up features (i.e. addition of manufacturing information) in the KBE application has been given only limited attention as it is the least complex process step to automate. However, a select number of dress-up features have been implemented to illustrate the feasibility. An illustration of a formboard generated with the f3fit application containing dress-up features is shown in figure 6.40.
Generate manufacturing symbols  The f3fit HLPs are implemented with both a fitting and a dress-up visualization mode. This includes representations of formboard symbols, in the fitting mode as simple outlines and in the dress-up mode as the actual symbol.

- **bundle** HLP: the fitting mode distinctly represents the bend-sections and flipping segments (see figure 6.7), while the dress-up mode shows an outline with a dashed centerline (see figure 6.40).

- **endpoint** HLP: In both modes the endpoint components are represented by their outlines. Clocking symbols, cutting lines, tolerance fields and compo-
6.6 Discussion: in search of a search method

Component codes are symbols associated with the HLP. The last two have no representation in the fitting mode while the first two are represented by an outline. In the dress-up mode the outlines are hidden and the actual symbols or texts are placed. Figure 6.41 shows the class diagram om the endpoint HLP.

- connection-point HLP: For fitting break-out connection points are represented by a circular plane while no geometrical representation is given in the dress-up mode. However, symbols are associated with the connection-point as well, such as a tolerance field and destination code text fields. Destination codes are used to specify the route a wire must take from one endpoint to the next. The destination code fields are filled by relevant numbers by evaluating the `set-destination-codes` functions in the `main-branch` and `branch` SCNs. This algorithm is presented in figure 6.42. The main requirements are to use codes consisting of three digits and one letter (e.g. 123A), start at the left side of the main branch with 010 and use codes rounded up to tens for the main branch codes (e.g. 020).

Generate legend, rulers, etc. The formboard-frame HLP is represented as an outline during fitting. In the dress-up mode the outline is hidden and detailed features that are not directly related to one of the wiring harness HLPs are added such as rulers, a legend box, change notice box and identification boxes.

Export drawing to pdf The formboard-drawing CM, indicated in figure 6.32, generates a drawing of the fitted, dressed-up wiring harness and formboard frame. Several functions are available in fitting-container-branches to export the drawing to PDF. Other formats are available as well, although these have not been thoroughly tested: DXF and STEP.

The f3fit application This section has presented the implementation of methods discussed in sections 6.2 and 6.3.2 into a KBE application. Like the f3flat application, a Graphical User Interface (GUI) is developed for the f3fit application to enable inspection and manual adjustment of the fitted wiring harness configuration, see figure 6.43.

6.6 Discussion: in search of a search method

This chapter has presented approaches to and an implementation of automating the formboard fitting and addition of manufacturing information (i.e. dress-up) process steps. This manual, multi-disciplinary design problem was automated by
Figure 6.41: Class diagram of the \textit{f3fit} endpoint HLP.
6.6. Discussion: in search of a search method

**Figure 6.42:** Activity diagram and example of the *set-destination-codes* algorithm, as implemented in the *f3fit* application.
implementing rules for analysis and checks in the \textit{f3fit} KBE application and applying search techniques to develop a manufacturable solution. The \textit{f3fit} geometry model is constrained to the limits set by the analysis of the wiring harness 3D model, removing the need for post-fitting checks.

As no optimization or search problems were identified that were directly comparable to formboard fitting, different approaches were experimented with. The genetic algorithm meta heuristic approach shows promise, but improvements are needed in the objective formulation and solving speed. Configurations generated with this method were significantly dissimilar to manually-made formboards. In parallel the alignment method was developed, which performs fitting steps on a wiring harness based on manual solving heuristics. The method does not always find a feasible solution, requiring (minor) manual adjustments for certain harnesses. It does generate configurations more closely resembling manually created drawings, thereby creating confidence in the approach.

Many improvements of the described fitting methods and alternative approaches can be thought of. The alignment method could be improved by having branches remember more feasible strategy combinations instead of only the best one as an-
other strategy may lead to a better global configuration. Other examples of improvements are the addition of different strategies, a horizontal alignment function, meta heuristics to select strategies, etc. With so many directions, it is recommended to develop improvements or alternative approaches based on experiments with the developed \textit{f3fit} tool set to ensure that these solutions are relevant to wiring harness development industrial practices. Manual fitting should be considered as a viable alternative to automated fitting: a fully constrained model can be operated on by low skill personnel with little applicable knowledge. These workers could even be found via crowdsourcing [125] where fitting work would be presented as an online game or via crowdsourcing platforms such as Amazon’s Mechanical Turk [126].

Fitting was described as a multi-disciplinary optimization problem, the proposed alignment method however considers the different disciplinary aspects sequentially. The mechanical flexibility is decoupled from the fitting steps by the bend assignment and analysis parameterization approach. Flexibility is incorporated as angle limits in the \textit{f3fit} model. The objective of limiting table size (floor space) conflicts with providing space around components (ergonomics) and limiting the amount of bending (install time, length variation). These are dealt with separately by first getting a tight fitted configuration without crossings, setting the smallest frame size possible and then making better use of the space within the frame to achieve an ergonomically more favorable layout. A method to ergonomically score a formboard configuration was developed, but not fully implemented. Further testing is required to validate the method settings and implement it in the formboard layout fitness function.

Manufacturing symbols (i.e. dress-up features) on the formboard can be standardized in order to ensure uniformity on the manufacturing floor. In cases where customers desire specific symbols on the drawing, for example for quality control, switching to alternative symbol representations could be made possible. The implementation of dress-up features is relatively straightforward, the features presented in this chapter were implemented in a single week. Not all symbols can be standardized, for these cases placeholders can be defined in the KBE application and they can be added using a CAD system (e.g. CATIA, AutoCAD) or other (lower-cost) graphics systems (e.g. MS Visio, Adobe Illustrator, Inkscape). The use of standard bend radii on formboards presents another opportunity: instead of adding multiple pins along the bend, a standard bend mold could be used, thereby reducing the time needed to set up the formboard drawing.

To conclude, a demonstrator application to perform both fitting and the addition of manufacturing symbols was developed. The resulting harness configu-
rations are acceptable to formboard engineers as layouts resemble existing formboard drawings or only need minor manual adjustments to do so. To quote one manufacturing engineer, in Dutch, "Die had ik ook kunnen maken" (I could also have made that one).
Chapter Seven

Process integration hurdle: CAD exchange

The automation solutions presented in chapters 5 and 6 must be integrated in the overall wiring harness development process in order to provide a complete solution. This requires interfaces between the preceding and succeeding processes, where the former presents a notorious issue in concurrent engineering: Computer Aided Design (CAD) model exchange. Aircraft integrators typically mitigate most CAD model exchange issues by requiring their suppliers to use the same CAD system as the integrator. But as the manufacturers do not use the same system, suppliers are subsequently confronted with diverse CAD systems.

The input required by and the output desired from the formboard design applications is presented in section 7.1. This section also introduces the actual file formats that can be expected as inputs, revealing the need for a CAD exchange method. A brief introduction in CAD exchange practices is given in section 7.2, including neutral standard formats for CAD in general and electrical wiring specifically. A solution for importing wiring harness 3D CAD models is proposed and implemented as described in section 7.3 using STEP and a KBE application to generate a standard definition. Section 7.4 finally discusses the resulting KBE supported formboard design process.

7.1 Input and output requirements for formboard design

The 3D design phase determines the available input formats and the manufacturing phase sets requirements on the output format(s). The main output requirement is that the formboard drawing is exported to a format that can be sent to a plotter and printed. This requirement is met by exporting the drawing to PDF, as was presented in section 6.5.3. The inputs of the f3flat and f3fit applications consist of parameter lists for each HLP. These lists can be provided to the applications directly or they can be read from a .dat file. It is envisioned that these parameters will be included in the EWIS Product Data Management (PDM) system. Figures 1.1 and 1.4 provide an overview of the input files and figure 7.1 illustrates some geometrical parame-
ters required for the \textit{f3flat} application. NURBS is the standard method to describe curves in most CAD systems \cite{127}, and it is therefore used to describe the bundle centerline geometry. The formboard design applications do not require endpoint components with detailed features, an abstraction is sufficient (see section 6.4.2)\textsuperscript{1}.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure7_1.png}
\caption{Illustration of the input and output formats provided or required by the various design tools.}
\end{figure}

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure7_2.png}
\caption{Examples of wiring harness modeling differences.}
\end{figure}

Engineering companies use different CAD systems and design conventions leading to a variety in supplied 3D design formats, as indicated in figure 7.1. Many com-

\textsuperscript{1}A link to the original 3D geometry can be saved in case the detail geometry is needed at a later stage. For example to create 2.5D blocks for a non-flattenable wiring harness.
panies in the aviation sector use the CATIA V5 CAD system (even releases of this single system are often not compatible), but legacy systems are also still in use such as CATIA V4. Other systems as used as well, such as Siemens NX, CADDS5, CATIA V6 and AutoCAD, and new systems can be expected to be used in the future. Besides the native formats of these systems, in some cases neutral formats such as IGES and STEP (see next section) are used to provide the 3D design.

The conventions adhered to during a 3D design process will differ as well because the process is performed by different teams from different companies and the electrical wiring workbenches provided by the systems differ. This leads to significant differences other than the file format between the models provided for manufacturing design. The main differences found by studying a variety of 3D models from Fokker Elmo projects are in the naming of entities, geometry conventions, structure/topology of the product tree, division in zones and availability of non-geometrical data (connectivity data, annotations). Figure 7.2 illustrates two typical topology differences that were found in the 3D models.

The model differences lead to the need to either use multiple CAD systems, or convert provided designs to the company standard format. The latter is not always trivial and often results a large amount of repetitive manual work. The next section explores the options for CAD exchange, as a solution is needed in order to integrate the KBE supported formboard process in the overall Electrical Wiring Interconnection System (EWIS) design process.

7.2 CAD exchange

Interoperability between CAD systems is an issue in many multi-disciplinary product development projects involving multiple suppliers, with different geographical locations [127]. Accurate data exchange is recognized as a key concept for concurrent engineering [128]. The costs of CAD data interoperability issues are often indicated to be in the order of billions [129]. These indications tend to refer to a single study from 1999 [130] in the automotive industry however, which may be dated. Xu [127] notes that the costs of data translation are often hidden. The pertinence of this issue to industry is probably better reflected in the large number of companies offering CAD exchange and repair (i.e. ‘healing’) services.

Inconsistencies between geometry data formats come from differences between...
7. PROCESS INTEGRATION HURDLE: CAD EXCHANGE

the (most often) proprietary data formats used by the CAD systems and the CAD kernels they are built on. CAD systems are built on geometry modeling kernels, which are software packages implementing mathematical functions for geometry modeling\(^3\) [127]. The CAD system application interface interacts with the kernel in order to create geometry models. Most CAD systems are feature-based and history-based, depending on proprietary data structures. The differences between systems lead to data issues, of which examples are given by Xu [127]: aggregate errors such as zero-volume parts, duplicates, missing parts; and numerical imprecision errors such as cracks or geometry overlaps. Non-geometrical inconsistencies are a problem as well, for example annotations are often not or not properly translated, requiring time-consuming model checks and adaptations.

7.2.1 CAD exchange methods

One approach to deal with geometry kernel inconsistencies is to use multiple CAD kernels in a single CAD system. CAXA (www.caxa.com) is an example of such a **dual kernel CAD system**. Xu [127] indicates that this only solves the problem partially as there are many other kernels and he observes that these CAD systems are difficult to build.

With **direct data translation** a model is translated directly from one CAD system to another. Such a translator may be provided by the CAD system itself. In many cases a neutral database is set between the different CAD systems \(^4\). This has the disadvantage that the chances of translation failure increase when models become more complex [127]. An alternative direct translation option is to develop a (domain-specific) custom application using the Application Programming Interface (API)'s of the CAD system itself (e.g. VBA or C++ scripting in CATIA V5).

A third option is to use **neutral intermediate formats**, which can be proprietary or open. Some of the most widely adopted formats are DXF/DWG, IGES and STEP. The 2D drawing format DXF is an open version of the proprietary AutoCAD DWG data format [127]. The Initial Graphics Exchange Specification (IGES) format was developed in the 1970's for surface modelling and is still supported by most

---

\(^3\)Some examples of kernels available -through licensing- are ACIS used by AutoCAD (www.autodesk.com) and SpaceClaim (www.spaceclaim.com); Parasolid used by Siemens NX (www.plm.automation.siemens.com/en_us/products/nx); CGM used by CATIA V5 and V6 (www.3ds.com/products-services/catia) and SMLib used by GDL (www.genworks.com); OpenCASCADE (www.opencascade.org) is an open-source kernel.

\(^4\)There are a number of companies offering data conversion services such as Datakit (www.datakit.com), Theorem (www.theorem.co.uk), Capvidia (www.capvidia.be) and Coretechnologie (www.coretechnologie.de). Of these, Datakit offers an interface with Smlib, the geometry kernel used by GDL.
CAD systems. IGES was expanded with technological and organizational data to the PDES format, which led to the development of the ISO 10303 Standard for the Exchange of Product model data (STEP) standard [127]. STEP is now the main industry standard and aims at supporting data exchange, sharing and archiving. An Application Protocol (AP) is an implementable data specification for STEP, addressing specific products and processes. Two APs that most commercial CAD systems are able to export and import are the STEP AP203 'Configuration controlled 3D designs of mechanical parts and assemblies' [131] and AP214 'Core data for automotive mechanical design processes' [132] formats. These parts cover the definition of mechanical components and document purely geometric information.

The use of neutral formats comes at the cost of information loss. When CAD data is exchanged to a neutral format higher-level data such as features and the history tree are not transferred. To recuperate features the model must be re-mastered. Vergeest [133] states that the loss of the design intent when transferring CAD data poses a fundamental restriction of interoperability. According to Vergeest, this should be recognized and compromised solutions should be developed intentionally, instead of adopting ad-hoc workarounds.

More details on CAD exchange issues and technologies can be found in [127] and [133]. Exchange approaches are being researched such as neutral API, procedural model data exchange [128]. An overview of specific STEP APs relevant for the aerospace industry is given in [134] and an example of its application at the Boeing Company is given in [135]. An interesting research effort aiming at including functional and behavioral information in STEP (i.e. semantically enriching the product definition) is presented by Barbau et al. [136], who translate the STEP schema to the Ontology Web Language (OWL). More domain-specific standards (often, but not always STEP APs) are being developed by industry. In construction for example, neutral Building Information Model (BIM) formats are developed (e.g. IFC). The next section introduces available standards applicable to wiring harness design and manufacturing.

### 7.2.2 Wiring exchange standards

STEP APs have been developed for multiple domains, including STEP AP212 'Electromechanical design and installation'. This extensive standard specifies data representation for electro technical plants and industrial system design information [137] [134]. No commercial CAD or wiring harness design tools were found supporting this STEP part.

The German Verband der Automobilindustrie (VDA) and ProSTEP iviP Associ-
ation jointly developed a derivative of STEP AP212 intended to address data exchange for car electric development, the Vehicle Electric Container (VEC) [138]. The VEC aims at supporting the entire electric system, it features the following subsets:

- Harness description list (KBL), for the exchange of wiring harness data (the harness, variants and modules, components, parts lists, connectivity lists, topology).
- Electrological data (ELOG), for the exchange of connectivity data (schematic diagrams, sheets and layout).
- Component model (KOMP), for the exchange of components data (component definitions).
- Geometry Model (GEO), for the exchange of geometry data (topology, 3D harness, 2D harness).

The geometry model includes many entities also used in the \textit{f3flat} input (figure I.1) such as the 3D bundle centerline, covering positions and component position and orientation. Not all parameters used in the KBE applications input files are accounted for however (e.g. endpoint flattening planes as in figure 5.19, flexibility parameters); if adopted the standard should be extended. No EWIS development tools are known to the author that use the full VEC data structure, the KBL subset however is used by various electrical design tools such as CHS by Mentor Graphics (www.mentor.com) and LDorado by Comsa (www.comsa.de). The authors of the VEC specification [139] note that the standard addresses the needs of the automotive industry, but may do so as well for the aerospace industry.

### 7.2.3 CAD exchange approach

A KBE application automating a detail design or manufacturing design process will in most cases require input geometry data which is supplied by different companies, using different CAD systems and conventions. An approach is needed to obtain the parameter set required to instantiate the KBE application HLPs, taking into account the differences in 3D models. Before proposing an approach for CAD exchange, the different options are discussed:

- Direct extraction with a custom-made application using the API of a CAD system. The advantage of this approach is that the precise geometry parameters can be extracted as well as non-geometrical attributes. The main drawbacks
of this approach are licensing and development costs and possibly the limitations of the API to extract the required data. Custom translators would be required for each CAD system in use.

- Direct extraction from one CAD system to another. This is most often done using a 3rd party extractor; the non-proprietary-neutral database puts a limit on the data exchange. These direct data extractors often do not account for information (features) provided by the CAD systems domain-specific workbenches for example. No direct extraction has yet been implemented in the GDL KBE system [53]. Direct extraction capabilities from the SMLib kernel [115] to other CAD systems/kernels and back are offered by Harmonyware [140], using DATAkit⁵ translators in some cases.

- Using neutral formats. Wiring standard formats (AP212, VEC) are not generally supported by CAD systems or CAD extraction tools. IGES and STEP (AP203 and AP214) are exported by most CAD systems however, where the former is being replaced by the latter. As with direct data extraction via neutral databases, (non-geometrical) information is lost. But as it is an open format, it is clearer what information is actually lost in translation. Note that information loss can occur at every translation step, but most is lost during the conversion to STEP. Different STEP writers may provide different content and structure to a STEP file: differences were observed between STEP AP203 files generated by CATIA V5 and NX7, provided by Fokker Elmo. Then there is also information loss caused by a STEP reader, which is not always representing all the data available in a STEP file. The GDL KBE system [53] is able to import and export STEP AP203 and AP214 files using a Harmonyware [140] STEP to SMLib translator⁶.

Method selection In practice, ad-hoc solutions are often developed to deal with CAD exchange issues, and each of the approaches discussed above are used, including complete manual re-creation (i.e. copying) of a design in a different CAD system. Manual design re-creation actually consists of very repetitive, rule-based activities and is often performed starting from a neutral format 3D model (e.g. IGES or STEP). These activities may be captured in a Knowledge Based Engineering (KBE) application. A degree of information loss can be expected with every CAD exchange solution presented. It is not certain that direct translators will be able to provide

---

⁵(www.datakit.com)
⁶Initially only a flat geometry list was imported from the STEP file to the GDL environment. Support was provided to Genworks in order to develop and test an improved STEP reader that imports the assembly structure, entity names, colors, etc. An improved STEP writer was developed as well.
more reliable data than a neutral format in all cases. For practical reasons using STEP AP203 or AP214 seems to be the most convenient CAD exchange approach. The main reason is availability: all major commercial CAD systems are able to read and write these formats, as is the GDL KBE system. Current CAD systems do in most cases not support domain-specific standards such as STEP AP212 and VEC for the wiring case. Differences in design definition formats between current and future project or CAD systems cannot be predicted. It is reasonable to expect that exchange methods will have to be developed or modified per project and/or CAD system. Current functionalities to automate the addition of manufacturing information at Fokker Elmo are often project-specific and limited for use to a specific release of the CAD system. It is clearly undesirable to have to redevelop the formboard KBE applications (f3flat and f3fit) for different projects and CAD systems as this negatively impacts cost, maintenance and quality assurance. To limit the impact of customer or project changes, the formboard process KBE applications and the solution of CAD import are designed as separate modules. This leads to the following approach.

**Proposed approach** The proposal is to export a design in STEP AP203/214 format from a CAD system, import it in the KBE system environment and derive the High Level Primitive (HLP) parameters from the imported geometry. This derivation, performed by a separate ‘interpreter’ KBE application, identifies assemblies, components, features based on geometry and topology rules and creates a parameterization matching HLP input parameters. These geometry and topology rules are specific to the 3D design product of the CAD system electrical workbench and/or customer project. The approach has the characteristics of rule-based feature recognition. In this case the aim is not only to identify features (fillets, holes, etc.), but also assemblies, components, geometry features and relations between entities and map these to the HLP model. Especially in cases where different conventions are used, analyses and geometry or topology modification steps may be necessary. The steps performed by the interpreter KBE application are based on steps taken in feature recognition [127]:

1. **Identification**: Entities (e.g. assemblies, parts, features) representing or relating to HLPs are identified within the imported STEP model. Identification of entities can be done based on geometry features, naming, assembly structure, etc.

2. **Parameter extraction**: The identified geometric entities are analyzed and geometric parameters are evaluated corresponding to the HLP product struc-
ture ontology. This can in some cases include modifications to the imported model.

3. **Organization**: Entities are named and organized in a parameterization of individual HLPs. HLP parameters can be collected in structured lists and/or in HLP instances.

4. **Inspection**: The imported model and the HLP-based output model are compared to validate the transformation.

Dependencies can exist between these steps for different model entities. The rules governing the identification of one entity may depend on the extracted model of another entity for example. The aim is to develop generic functions and classes that can be reused to develop interpreter applications for STEP files generated in different projects. The next section shows how this approach was applied to develop a STEP interpreter for the formboard design KBE applications.

### 7.3 Wiring harness model import

An interpreter application was developed to import a 3D wiring harness model designed in the electrical design workbench of CATIA V5, via STEP AP214 files. The application, called *f3input*, was developed using CATIA V5 files from a single aircraft project to test on 7. The native CATIA files were exported to STEP AP214 using the CATIA STEP export function. To ensure that not only visible Boundary Representation (B-rep) geometry but construction geometry was transferred as well, full export settings were used (e.g. including non-visible geometry).

#### 7.3.1 Wiring harness reparameterization

An overview of the general activities required to reparameterize a wiring harness model using the *f3input* application is given in figure 7.3. After exporting native CATIA files to STEP AP2014 these are imported in the *f3input* application Structuring and Collating Node (SCN) step–import (the *file-names* input-slot). The class diagram of the step–import SCN is given in figure 7.4. A sequence of assembly–import classes represent the contents of the input STEP files. The rules and methods to perform the HLP recognition steps (i.e. identify, extract, organize and verify) are implemented in the step–import SCN and the HLP SCNs. The KBE system allows flexible implementation of rules based for example on entity naming, topology and geometric features as will be explained next.

---

7Native CATIA V5 files of a current commercial aircraft project were provided by Fokker Elmo.
The first **identification** step (step 1) consists of identifying zone assemblies, as shown in figure 7.3. As explained in section 2.2.1, input wiring harness can be composed of zones, which need to be merged in a single wiring harness definition. In the test models, zones were identified as the first assembly node in a tree with an assembly name string not equal to nil. The identification of the wiring harness ID string is performed by extracting the common element from the zone assembly name strings.

In each zone, endpoints, bundle and covering assemblies are identified based on assembly name string matching. In each zone assembly, a breadth-first search is performed for nodes with an assembly string that matches a string from a list of
Figure 7.4: Class diagram of the *f3input* application SCN step-import.

(predefined) reference strings for assemblies of endpoints, for example: ("elc"). All the identified assemblies with endpoints are passed on to the endpoints SCN for extraction of the HLP parameters. The bundle and covering assemblies are similarly extracted based on lists of reference strings as well. The parameter extractions (step 2) are further detailed in sections 7.3.2, 7.3.3 and 7.3.4.

The extracted parameters are organized (step 3) by uniquely identifying them and collecting the parameters per HLP instance in plists (i.e. key value pair lists). These lists are collected from the HLP SCNs by the step-import SCN in a format readable by the *f3flat* application that can be exported to an output file (i.e. an *f3flat* input file, given in appendix I).

To ensure the correctness of the reparameterized model, it is instantiated together with the imported STEP model geometry and inspected (step 4). This is necessary because identification and extraction rules are depending on a number of CAD-system and project specific assumptions. More details are given in the following sections.
7.3.2 Bundle HLP reparameterization

The recognition steps for the bundle HLPs are shown in the activity diagram of figure 7.5. Unless otherwise specified, these activities are all implemented in the bundle-import SCN (see figure 7.4). Within the identified bundle assemblies, the geometries of the bundle centerlines and the circular curves representing the bundle diameter must be identified. This is again done by name-string matching, resulting in a list of circular curves and a list of centerline curves.

For each circular curve, the corresponding diameter is computed. The centerline curves feature overlapping curves and curves that do not match the definition of a bundle from one break-out point to the next (see right side of figure 7.2). A function was implemented to remove or trim overlapping curves and merge or split curves in order to obtain centerlines with start and end points at break-outs or endpoint connections. A unique ID is assigned to each resulting bundle centerline. For each centerline, the ID’s the centerlines connected to it are identified. The overview of connections between bundles provides the input to evaluate whether the imported wiring harness features a closed loop (i.e. a graph cycle).

The right side of figure 7.2 indicated different manners in which bundles can be modeled within an endpoint component. As the path of the bundle within an endpoint is not important to create a formboard, bundle centerlines or sections enclosed by an endpoint are removed or trimmed. This requires the extraction of the endpoint components as described in section 7.3.3. The remaining (final) bundle centerlines are matched to one of the circular curves to determine the bundle diameter. This is done by searching for the circular curve with a midpoint on the bundle centerline.

After performing the activities described above, all the bundle HLP 3D definition attributes (i.e. centerline NURBS definition, diameter, ID, connected component IDs) have been determined. The HLP parameters are collected in lists to be included in the $f3input$ application output. To verify that the geometry has been extracted correctly, an instance of each bundle is generated (the $f3input$ bundle HLP) to compare their NURBS centerlines to the bundles B-rep representation from the STEP file(s).

7.3.3 Endpoint HLP reparameterization

Individual endpoint assemblies are identified in the different assemblies with endpoints, provided as input to the endpoints SCN. The identification is performed by means of name-string matching (i.e. searching for assembly names "endpoint"
or "end") and/or evaluating whether a child node of the endpoint assemblies contains B-rep geometry. For each identified endpoint assembly, an endpoint HLP is instantiated from which parameters can be extracted. The corresponding activity diagram is given in the top-left of figure 7.6. The structure of the endpoints SCN is shown in the class diagram of figure 7.7.

**Endpoint extraction**  Given the endpoint assembly, the endpoint ID can be evaluated by the endpoint HLP. Identical assembly name will often be used for harnesses constructed in different zones. In order to uniquely identify each endpoint, the zone name is added to the assembly name (e.g. 'Endpoint-001-zoneA' and
7. **PROCESS INTEGRATION HURDLE: CAD EXCHANGE**

Figure 7.6: Activity diagram of the ‘extract endpoint parameterization’ activity of figure 7.3. *depends on bundle parameters, see figure 7.5.*

---

[Diagram showing the process steps]

1. **Identification**
   - identify unique endpoint assemblies
   - collect all endpoint parameters
   - extract endpoint parameters

2. **Parameter extraction**
   - identify component assemblies
   - determine component types
   - determine connected bundle
   - check if connecting bundle is enclosed by endpoint
   - extract connector parameters
   - backshell present?
   - bundle extraction*?

3. **Organization**
   - determine component geometric parameters
   - collect component parameters
   - visualize original B-rep and abstracted geometry

---

*Figure 7.6 is an activity diagram for the 'extract endpoint parameterization' activity described in figure 7.3. It illustrates the process steps and dependencies involved in extracting and organizing endpoint parameters. The diagram includes steps for identifying unique endpoint assemblies, collecting all endpoint parameters, and extracting endpoint parameters. Similarly, it outlines the process for identifying component assemblies, determining component types, and checking if connecting bundles are enclosed by endpoints. The parameter extraction phase involves determining connected bundles and extracting connector parameters. The diagram concludes with the organization phase, including determining component geometric parameters, collecting component parameters, and visualizing original B-rep and abstracted geometry.*
7.3. Wiring harness model import

Figure 7.7: Class diagram of the endpoints SCN as implemented in the f3input KBE application.
Endpoint-001-zoneD’). Endpoint component assemblies are identified by selecting the children of the endpoint assembly from the STEP model and determining whether they represent a connector or backshell. In case only one component assembly is present, it is a connector. In case multiple component assemblies are present, the selection is based on comparing name-strings. A warning is given to the user when components are found that cannot be identified. As shown in figure 7.7, a connector and -when present- a backshell class are instantiated with as input their respective assemblies from STEP. These classes extract the connector and backshell parameters. To determine which bundle connects with the endpoint, the bundle centerlines are provided as input and subsequently compared to the endpoint geometry. Bundles that are fully enveloped by an endpoint are identified and provided to the bundle-import SCN as input (see figure 7.5).

**Connector extraction** The connector class identifies B-rep geometry, curve and point geometry nodes in the STEP connector assembly. Figure 7.7 indicates that different connector child-objects can be generated. Which of these types is selected depends on a set of rules testing whether characteristic points (e.g. bundle connection point) and lines (e.g. keyway vector) are present and functions that determine if a component is circular or not. The determine-if-connector-is-circular function selects the mating face of the connector (i.e. the face that connects to another connector) and tests its edges to identify whether is is circular (only arc curves) or rectangular with fillets (straight and arc curves). In order to find the connector mating face, the direction of the y-axis is determined from the connector B-rep local reference frame.

Selecting a connector type generates an instance of that particular class. In case the connector is circular, an instance of the connector-circular class is generated, which mixes in the connector-mixin class. The latter requires the B-rep geometry and the reference axes as inputs, which are used to evaluate computed-slots that determine the abstracted parameterization of the component. For the circular connector, the parameterization includes characteristic points (front and back points), the component centerline, the maximum radius and the keyway angle. The extracted connector parameters are collected in lists.

Figure 7.6 does not show the activity diagram for backshell components, which is nearly identical to the connector component activity diagram. The main differences are that there are less backshell types and that the component can be angled.

Using the KBE system Graphical User Interface (GUI) both the STEP geometry and the extracted geometry of components can be visualized. This is illustrated
7.3. Wiring harness model import

in figure 7.8 for an endpoint with a connector and an angled backshell. In case an identification or extraction step is incorrect, settable slots allow a user to modify the assumptions or parameters (e.g. connector type, relative axes definition). This was sometimes necessary as the connector type identification rules are not sufficiently robust to reliably identify each component. A second inspection can be performed by generating an extraction report (see figure 7.8) and have a wiring expert review it. The report is generated by the import-report-document Capability Module (CM), see figure 7.4.

![Using a Graphical User Interface](image)

**Figure 7.8:** Inspection options for endpoint assemblies. Left: screenshots of the f3input application. Right: sheets of an export report automatically generated (PDF) by the f3input application.

7.3.4 Covering HLP reparameterization

The coverings SCN generates a covering HLP for each covering assembly from the STEP input (see figure 7.4). The covering ID is obtained from the assembly name. From the assembly characteristic points, the covering centerline and the B-rep geometry are identified (using name-string matching). In most cases the covering centerline is larger than the section covered by the B-rep geometry. For the tested models, points were present in the model delimiting the covering ends. After identifying these points the covering centerline is trimmed. The covering centerline is then compared to the bundle centerlines for overlaps and the relative position of the covering with respect to covered bundle centerline(s) can be determined. The relative parameters are also stored in lists according to the f3flat input file structure. The resulting covering parameterization can be inspected in the KBE system GUI by visualizing both the STEP input B-rep and the reparameterized geometry.
The \textit{f3input} application was developed with reference models from a single project. Models from two other projects were imported as well. This was successful in one case, where the wiring harness model was designed using the same CAD system (CATIA V5) and unsuccessful for another case where the Siemens NX CAD system was the design tool. Note that not all component variants were implemented as the scope of the work was to provide a working demonstrator. The selection rules can be adapted for different cases. The larger the difference in product model and naming conventions, the more the application will need to be modified. Most identifications are based on naming conventions, it is therefore important that these are adhered to. With the \textit{f3input} application, the complete formboard design process can be supported as is discussed in the next section.
7.4 Discussion: the KBE supported formboard process

To integrate the formboard design applications in the EWIS development process, input and output must be provided and generated matching the existing preceding and succeeding processes. Importing wiring harness design data presents the main challenge because of the large variety in design definition formats with different projects or customers. The differences between CAD system wiring workbenches and design teams make the outcome of the design process difficult to predict. This behavioral and structural complexity of the 3D design process increase the nested complexity of the formboard design process. Variation in CAD definition formats is a recurring problem in concurrent engineering.

As a practical solution, a KBE application was proposed to reparameterize geometry from a static neutral format. The $f3input$ application imports a STEP AP214 file of a wiring harness generated by a CAD system and then identifies parts, extracts parameters, organizes parameters by HLP instance and allows inspection if the transformation. Many of the assembly or component recognition steps could be simplified if a higher level of standardization would be adhered to during 3D design. In practice, the variation in 3D models cannot be expected to reduce much. Even if wiring standards such as VEC become more mature and are adopted by industrial players for new aircraft projects, older aircraft programs will still provide a large variety in CAD systems. An aircraft program can easily run for 30 years and changing to newer CAD systems will only occur if the cost and configuration management risks of switching outweigh not doing so. In practice, it should be anticipated that a KBE interpreter application must be set up or adapted for each aircraft project. Implementations of interpreter variants could reuse many $f3input$ components. The extent in which these components can be re-used was not investigated and proving reusability would require application to multiple different projects.

The exchange of design data between the KBE applications is done via simple .dat files or plists. Keeping the exchange format simple provided flexibility during development and limits difficulties switching to an external database in a future stage. When industrialized, the 3D and 2D wiring harness parameters could be integrated in the company configuration management database. This integration was not within scope of this research. The output of the formboard design process (generated by the $f3fit$ application) is a 1:1 scale PDF document of which a few were sent to a plotter and successfully printed. Other output formats are available as well (e.g. DXF, SVG, STEP).

---

8At Fokker Elmo this is the Wiring Design and Manufacturing System (WDMS).
For a KBE solution to be successful it needs to be integrated with preceding and succeeding processes. The exchange of CAD data plays an important role. This exchange may become easier in the future with more open CAD systems and standardization. This may take a long time without strong incentives for the CAD suppliers to bring about this change. While such open systems are not available, approaches such as presented in this chapter can provide a solution. A KBE system can provide the means to implement domain-specific component and feature recognition rules to transform a static geometry model to a flexible parameterization.

Figure 7.10: The KBE supported formboard process. From a CATIA V5 3D wiring harness design to a printed formboard drawing.

With the $f_{input}$, $f_{flat}$ and $f_{fit}$ demonstrator KBE applications, a complete tool suite is available to support the automated generation of formboard drawings, see figure 7.10. The same sequential way of working as the existing formboard design process can be used, but not necessarily. The applications could be made available as engineering services, performing the transformations and analyses without direct human supervision. Formboard specialists would receive notifications after a batch of harnesses has been analyzed and can act on presented results. Similarly, fitting operations could be performed before presenting (multiple) results to an operator who can then decide to apply adjustments manually. This could increase the efficiency of the formboard design process and make better use of specialists. The next chapter presents the results of using the formboard design KBE applications to create formboard drawings for a representative set of cases, provided by Fokker Elmo.
Chapter Eight

Results

The previous chapter concludes by presenting the integrated automated formboard design process. The question remains whether this process actually reduces the time required to create a formboard drawing of high quality (or even of higher quality than in the manual case). This chapter aims at answering this question based on qualitative and when possible quantitative test results.

A formal validation was performed with Fokker Elmo using four wiring harnesses from commercial aircraft programs\(^{1}\). Section 8.1 presents the validation setup, the tests and outcomes. As the four harnesses tested for the company validation were relatively complex and therefore did not represent the majority of harnesses, 21 additional tests have been performed on simpler, more typical harnesses\(^{2}\). These are presented in section 8.2. The success of the KBE solution is reflected in the on-going preparations for industrialization of the developed demonstrator applications, which is briefly described in section 8.3.

For reasons of confidentiality, the identification codes of the test wiring harnesses are removed and replaced by a single letter designation. All harnesses tested in sections 8.1 and 8.2 are harnesses designed for a modern regional airliner. All tests have been recorded on video in order to demonstrate the required time and to show the actions taken during each test.

8.1 Company validation tests

To evaluate the complete toolset of Knowledge Based Engineering (KBE) applications for formboard design, a formal validation was set up with Fokker Elmo\(^{3}\). The test harnesses were selected by the formboard experts independently from the au-

\(^{1}\)Four native CATIA V5 files were provided by Fokker Elmo, of which two were complex cases from different aircraft projects, as well as a medium and a simple case.

\(^{2}\)Native CATIA V5 files of a current commercial aircraft project were provided by Fokker Elmo.

\(^{3}\)This work was carried out as a part of a joint KE-works, Fokker Elmo and TU Delft ’F3 pre-industrialization’ project.
8. Results

Data and consisted of a simple, a medium and a complex model from a regional airliner program (models V, W and X) and a complex model from a different aircraft program (model Y). The evaluation comprises all the functionalities from the STEP interface, analysis, flattening, fitting, dress-up to exporting a printable drawing.

Before performing the tests and evaluating the results, formboard quality success criteria were defined by the formboard experts [141]:

1. Adequate spacing around connectors
2. Correct break-out angles of branches
3. Respected twist angle and bend radius
4. Logical fit of harness board

The first criterion is one of the formboard layout ergonomic requirements (see section 2.5). Criterion 2 and 3 address the flexibility requirements of the flat state with respect to the 3D state. The last criterion is a more qualitative assessment by the Fokker Elmo experts of the overall layout of a wiring harness on the formboard (e.g. efficient use of available space, limited use of bends).

Additional agreements were made on the methodology and parameters to use. To perform the flexibility analyses, parameters corresponding to bundles with nomex braiding were used because the parameters for open bundles were not available at that time. Only the inserts of a multi-insert connector were to be shown, ignoring the corresponding backshell. The formboard drawing would only show dress-up symbols that were implemented in the demonstrator applications. As was explained in section 6.5.3, this is easy to automate, however on order to reduce implementation effort limited to a few essential symbols. The fitting can be comprised of both automatic and manual steps.

The tests were performed by the author while recording actions on video. The validation was an open demonstration session with formboard experts and management present. During this session, the $f3input$ application was demonstrated, 3D analysis results were presented, the recorded test videos for flattening and fitting were played back and the resulting formboard drawings were provided printed at 1:1 scale.

**CAD exchange** The $f3input$ application converts the STEP file to the f3 standard input format. Because information is lost during the conversion from native CATIA
8.1. Company validation tests

files to STEP, the f3input application augments the input data as was detailed in section 7.3. Harnesses V, W and X correspond to the regional airliner project which the f3input application was built for, and therefore all three could successfully be read and converted. Figure 8.1 partly shows harness V in the CATIA V5 environment and the visualization made by the f3input application. The time required to convert the STEP to the f3 format (without inspecting every single endpoint) was 80, 125 and 139 seconds for harnesses V, W and X, respectively.

Model Y corresponds to a different aircraft project, and although the model was designed in the same CATIA environment as the other harnesses, there were differences mainly in the geometric build-up and naming. Two aspects were that not all centerline curves could be identified by name-string matching and that the algorithm to remove overlapping sections failed in certain cases. Because only a single harness of this aircraft program was to be tested, the conversion to the f3flat input file format (see appendix I) was performed largely manually, taking about 10 hours of work, instead of modifying the f3input application or developing a separate interpreter application. As stated in section 7.4, different CAD systems and/or aircraft programs will probably require the development of specific interpreter applications.

![Figure 8.1](image1.png)

Figure 8.1: The conversion from CATIA to the f3flat input format. Left: Wiring harness model V is loaded in CATIA V5R20 and exported to the STEP AP214 format (screenshot of CATIA V5R20). Right: The STEP AP214 file is loaded in the f3input interpreter application and reparameterized (screenshot of the f3input application).

Demonstrating the f3input application convinced the Fokker Elmo experts of the technical feasibility of the interface between the 3D design CAD environment to the f3 application environment [142]. As anticipated in section 7.4, custom STEP
interpreter modules will be required to cope with differences between 3D models from different aircraft programs. The experts concluded that 3D designers should adhere to standards in order to have more consistent outputs.

8. RESULTS

Analysis and flattening   The 3D analysis of models V and W show no bend radius violations or 3D break-outs. Some bundles are designated as potentially posing problems: this is the case for 5 and 9 bundles of models V and W, respectively. Inspecting these reveals that their length falls outside of the available flexibility parameter range which the analyses are based upon. These sections must therefore be inspected by an expert before flattening the model. Both the 3D and flat results can be observed in the top part of figure 8.2.

The analysis of models X and Y shows more serious problems. There are a large number of short bundles for which no flexibility parameters are available and must be checked manually. Several 3D break-outs are identified indicating that the harnesses may not be flattenable. Again a formboard expert must check these and possibly override the warning based on the provided out-of-plane angles and his/her experience. Model X features a bundle where the allowed twisting angle is violated. Harness Y shows a bundle section that cannot be flattened at all: a 3D bend. This effectively means both harness X and Y are not allowed to be flattened\(^4\). However, for the sake of testing the f3fit application, both harnesses are still flattened, as shown in the lower part of figure 8.2. Note that the 3D bend is flattened based upon its in-plane curvature distribution (see section 5.5.2). No comparison to the flattening result by Fokker Elmo can be made as only the fitted formboards are available.

The analysis performed is fully based on the existing process at Fokker Elmo, so twist angle and bend radii checks are performed and when necessary respected in the same manner. Break-outs are designated as 3D when the out-of-plane angle exceeds 10 degrees. This means that the application meets the success criteria of ‘respecting twist angle and bend radius’ and ‘correct break-out angles’, respectively. The Fokker Elmo formboard experts indicated that the results are satisfactory and that the use of these analysis functionalities would benefit their work [142].

Fitting and dress-up   Harness model V, a typical simple harness, was fitted automatically and then adjusted manually to align the long bundle with the x-axis\(^5\). The frame was set and dress-up symbols were added automatically, resulting in the con-

\(^4\)Flexibility parameters for nomex braiding were used for the analysis, which may be too stringent.
\(^5\)The fitting method aligns branches with respect to their parent branch, not with respect to the x-axis. Based on these results, the conclusion can be drawn that a providing a function to align bundles with the x-axis after fitting would be beneficial.
8.1. Company validation tests

Figure 8.2: 3D (left) and flat (right) views of test harnesses V, W, X and Y. Images created by the $f3flat$ application, all have different scales.
figuration of figure 8.3. The automatic computations for this step took 36 minutes and the manual adjustments only 3 minutes.

![model V](image)

Figure 8.3: Screenshot of the PDF generated by the f3fit application after applying fitting and dress-up to wiring harness model V. Intermediate steps are illustrated in figure J.1

In the case of test harness W the automatic fitting approach does not result in a feasible configuration\(^6\) and therefore the harness was fitted manually, the result is shown in figure 8.4. After obtaining a tightly-fitted configuration manually, the frame is set and the automatic relaxing function is used to improve the space around components and reduce the number of bends. In total 40 minutes were spent on fitting the harness manually and 27 minutes by the automatic functions.

![model W](image)

Figure 8.4: Screenshot of the PDF generated by the f3fit application after applying fitting and dress-up to wiring harness model W. Intermediate steps are illustrated in figure J.2

Harness X, designated as complex, was first fitted automatically and subsequently further fitted manually in order to obtain a better fit. Figure 8.5 shows the dressed-up result.

According to the Three Dimensional (3D) analysis, flattening model Y is not allowed because of the 3D bend that was identified. To demonstrate the fitting capabilities for this specific case the analysis result was overridden and the 3D bend sec-

---

\(^6\)The tests were performed with the bottom-up alignment strategy to limit the number of variants to consider and hence the calculation time. Extending the alignment procedure by iterating between branch levels would probably result in a satisfactory automatic alignment, at the cost of computation time.
8.1. Company validation tests

Figure 8.5: Screenshot of the PDF generated by the \textit{f3fit} application after applying fitting and dress-up to wiring harness model W. Intermediate steps are illustrated in figure J.3

This is not allowed during normal operations and overriding analysis outcomes should only be performed by senior manufacturing engineers and documented.

The flattened harness features many short bundles that cannot be flipped and clusters of endpoints close together overlapping each other. Except for the longer branches branching out, automatic alignment will not produce any results for this case. To obtain a fitted result without overlap, the only option was to modify break-out angles, which has to be done manually because of the absence of validated rules. The modifications of break-out angles needed to generate the fitted result of figure 8.6 are in many cases probably not allowed as the divergence from the 3D model becomes larger.

These test harnesses provided for validation could not be fitted completely automatically. Manual fitting was required and in the case of model Y some break-out angles had to be modified. The latter led the formboard experts to conclude that formal rules for adapting break-out angles should be defined. The break-out angles can be less restricted than in the current \textit{f3fit} implementation, according to the experts. Lacking formal, validated rules, modifying these angles must remain a manual decision however. Another observation was that engineers have more freedom in the current fitting process using CATIA V5 compared to the \textit{f3fit} application. Of course, the former requires time to be spent on checking the resulting formboard configuration with the 3D model. The formboard experts approved of the symbols generated as dress-up, especially the automatic creation of destination codes was appreciated. They noted that the positioning of certain endpoints could be improved though, for example by further separating them.
In terms of process time, significant manual activities must be performed for the medium and complex wiring harnesses, taking up to 50 minutes for test harness Y. A note is also made of the fact that time will be spent on activities that are not automated such as downloading CAD models, plotting the drawing and other administrative work. The formboard experts conclude that even with these manual activities significant time savings can be achieved compared to the current situation [142].

Several notes can be made on the validation procedure presented in this section and its results. The time available for the formboard experts was limited. Most aspects of the applications were already verified in prior sessions (i.e. knowledge verification meetings), in fact most functionalities of the applications were developed in close cooperation with the experts. This probably contributed to experts not explicitly stating that a feature/function was correct. There is also the tendency to propose further additions, implying but often not explicitly stating, that the delivered/presented functionalities are as desired. The involvement of most experts in defining the implemented rules and methods may also be a source of bias\footnote{At the start, the experts were very skeptical of the research project feasibility. Some explicitly stated that formboard design could not be automated at all. Their confidence in the viability of the KBE solutions increased each time a demonstration session was held. In the end their attitude was changed to enthusiasm and they became advocates of the solutions within the company.}. Another limitation of the validation is the number and type of test cases. The four wiring harnesses are not representative for the range of harnesses at Fokker Elmo. In practice only a few harnesses are classified as 'complex' and the majority as 'sim-
8.2. Application performance tests

The KBE applications analysis, flattening, fitting and dress-up results were according to the formboard expert expectations. They expressed confidence in the capability of the formboard design suite to create quality formboard drawings fit for manufacturing [142]. As the number of test cases was limited, more tests were performed on a larger (more representative) batch of wiring harnesses, as described in the next section.

8.2 Application performance tests

In order to obtain a better evaluation of the formboard design time reduction achievable by using the f3 KBE applications, a batch of 21 harnesses was tested. The 3D designs were provided by Fokker Elmo and all correspond to the same regional airliner as test harnesses V, W and X of the previous section. These harnesses are designated A to U and most were classified by Fokker Elmo as 'simple'. Some of the provided harnesses actually only feature a single bundle and two endpoints. These harnesses are not relevant to evaluate the fitting functions and are excluded from the f3fit tests. All tests were performed on a HP laptop$^8$. All actions were recorded on video in order to provide means of verification and record elapsed time.

CAD exchange The 3D files tested are in native CATIA format and must be converted to the f3 standard input. The f3input application can generate the f3 input file based on the CATIA file converted to STEP, as was detailed in section 7.3. The conversion to STEP has been done manually by using the STEP export function available in CATIA V5. It is also possible to do this automatically in batches however, for this reason the conversion from CATIA to STEP is not accounted for in the tests below. Typically the time to load a CATIA wiring harness and convert it to STEP will be from a few seconds to a few minutes.

Figure 8.7 shows the procedure followed to convert each wiring harness in STEP format to the f3 standard input. After loading the STEP file, a visual inspection of each endpoint component is required to ensure that the endpoint component type and orientation are correctly identified by the program. In a few cases the component type had to be manually specified or the orientation axes had to be switched (e.g. switch x and y axes). The need for these visual inspections could be reduced or eliminated by improving the consistency of the provided 3D models and by improving the component identification and extraction functions of the f3input application.

$^8$Dual core CPU 2.40 GHz (one core used by the application), 4GB RAM.
8. Results

Figure 8.7: Procedure used to convert a wiring harness in STEP format to the f3 standard format. Test harness: F.
8.2. Application performance tests

On average, the conversion from STEP to the f3 standard input took two and a half minutes. Figure 8.8 shows that all 21 test harnesses are close to this average except for wiring harness F. This is because harness F contains 45 endpoints and the visual inspection of all the endpoints is the most time consuming activity.

**Analysis and flattening**  The f3 standard input can be loaded into the f3flat demonstrator application for analysis and flattening of the wiring harness 3D model. The procedure followed is illustrated in figure 8.9. When loading the f3flat Graphical User Interface (GUI) all the analyses are automatically performed. Potential problem areas such as non-straightenable bends and 3D break-outs can be inspected in the GUI and the flattened model can be inspected. Finding the results from the analysis satisfactory, standard bends are assigned automatically to each bundle and the resulting 2D model is stored.

As indicated in figure 8.10, it takes four minutes on average to perform the analysis and flattening steps on these harnesses using the f3flat application. The amount of time required to load a model is shorter for models that are small in total bundle length and have fewer endpoints. This is illustrated by comparing models M and I,
Load the model, perform analysis

The $f3flat$ GUI is loaded in a browser window. All analyses are performed during the loading process.

Inspection of potential problem areas

The wiring harness product tree.
The inspector shows parameter values of the analysis results.
Components with potential problems are indicated in red.

Inspection of flat model

The analysis results can be inspected.

Flat state of the wiring harness model.
The flattened state of the wiring harness is generated.

Bend assignment & export for fitting

The $f3fit$ input file is stored in the ‘F3FLAT output’ folder.
The bend-assignment CM is called and an $f3fit$ input file is generated.

Figure 8.9: Procedure used to analyze, flatten and export a 3D wiring harness using the $f3flat$ demonstrator application. Test harness: F

210
which are both single bundle harnesses (i.e. only two endpoints). Model M with a bundle length of about 11 m requires 159 seconds for the analysis and model I with bundle length 0.8 m only 7 seconds. A server error occurred while loading model E (a single bundle harness with length 13m), which is expected to be the reason for the long time to load that particular model.

The time required for the visual inspection depends mainly on the number of issues found. For example, 13 issues are found in model F, all of which must be examined, leading to a total inspection time of 331 seconds. Models G, with 45 seconds to inspect, and model N, with 15 seconds to inspect, only present 6 and 1 issues, respectively.

The time required by the bend assignment functionality depends mostly on the number of bends, for which bending analyses are performed to determine the maximum allowed bend angles (see section 6.2.2). The larger the wiring harness, the more time these analyses require. It is therefore expected that the time required by the bend assignment Capability Module (CM) is similar to the time required by the initial loading and analysis step.\footnote{It would appear that this is not the case for harnesses A, B, C and D. This is not so however, in these cases the bend-assignment functions were already called while loading the GUI.}

Using the \textit{f3flat} demonstrator application, wiring harness analysis and flattening can be performed within a few minutes.

\textbf{Fitting and dress-up} The test harnesses have been fitted, dressed-up and saved to a PDF file using the \textit{f3fit} application. The single bundle harnesses are left out, since without any branches they are not relevant to evaluate the fitting method. The different steps used to obtain the final harness drawing are shown in figure 8.11. After loading the model, the alignment method (see section 6.3.2) is automatically performed. If needed, the resulting configuration is manually adjusted and the smallest possible frame size is automatically selected. To increase the distance between the elements of the tightly-fitted harness the orientation of angled endpoints is optimized and the bends are relaxed. Finally the switch to dress-up mode is made, destination codes and other dress-up symbols are automatically generated and the entire model is saved as a PDF document. Note that the exact sequence indicated in figure 8.11 was not always adhered to. Some cases were adjusted manually after setting the formboard frame for example. As explained below, harness drawings have been generated both fully automatically as semi-automatically. Some test harnesses required manual fitting actions before as well as after setting the frame size.
8. Results

Figure 8.10: Time required to analyze, flatten and export a 3D wiring harness model. The tests harnesses are sorted by number of connectors.

On average, it took 12 minutes per wiring harness to obtain the final formboard drawing, as shown in figure 8.12. For most of the test harnesses the required time was much less, ignoring harnesses F and P the average comes down to only 6 minutes. Harnesses A, G, N, R, S and U could be fitted completely automatically, without any manual work. The results can be seen in appendix J. Most of the time is needed to perform the automatic fitting iterations and again, the larger the harness, the more time is required.

Harnesses C and T could have been fitted fully automatically; the single adjustment applied to each (10 seconds) was not strictly required, but improved the layout. Harness N was fitted a second time, now with a manual adjustment after the full automatic fitting to reduce the formboard width. After automatic fitting, both harness L and Q feature a long bundle oriented out of the horizontal plane. These bundles must be manually adjusted before setting the formboard frame. Functions could be developed to prevent this, by for example selecting another starting bundle or applying a function to align bundles with the x-axis after aligning branches. The distance between two successive break-out points in harness H was too short for the bundles breaking out to be aligned without crossing. This had to be solved
8.2. Application performance tests

Figure 8.11: Procedure used to fit, dress-up and export a 2D wiring harness using the f3fit demonstrator application. Test harness: Q.
by manually adjusting break-out angles, which is not automated lacking formal, validated rules.

Model K features a cluster of short bundles and a single long one. This harness could have been printed to PDF without any fitting at all: all bundles are already in an aligned configuration, therefore automatic alignment is unnecessary. The long bundle was fitted manually to reduce the formboard width. Model S was fitted automatically in 8 minutes. It was found that it could be fitted manually in less time: 2 minutes.

From all the test harnesses, models F and P had to be completely manually fitted. Harness F contains a number of break-outs close together with fanning endpoints. The bundles of these fans overlap, which can only be overcome in the current implementation by modifying the break-out angles manually. Functions could be built to automatically fan these, as long as an expert checks whether it is allowed or rules are defined. Model P features a geometry similar to harness H: two bundles break out at a short distance from each other on the same side of a branch in opposite directions. Obtaining a solution without overlaps requires setting the break-out angles. Alternatively to setting break-out angles, the standard bend radius could be reduced: tighter bends might allow the application to find a feasible aligned configuration.

The time to fit harnesses F (34 minutes) and P (76 minutes) manually could be reduced dramatically with a better performing GUI. Currently the entire model is regenerated after modifying a single bend angle and must completely be reloaded in to the browser. This takes about 20 seconds in the first 5 minutes after loading model P, and increases to a full minute after 50 minutes of manual fitting. This is clearly not efficient, improving the graphical interface can therefore allow manual fitting in much less time than presented in figure 8.12. Eliminating this waiting time could reduce the total time needed to manually fit the harness at least by half. Another issue with the current interface is that the viewer re-centers to a fixed point after each adjustment, requiring the user to navigate and zoom to the area of interest again and again. The time for manual fitting could therefore be further reduced by reusing viewer settings.

Simple wiring harnesses can typically be fitted, dressed-up and exported for printing within minutes. In some cases minor manual adjustments are required, or preferable to obtain a smaller formboard frame. A few cases featuring overlapping geometries that cannot be solved without adjusting break-out angles require more manual work. The amount of time this requires can be further reduced with improved model viewing functionalities.
8.2. Application performance tests

Figure 8.12: Time required to fit, dress-up and export a 2D wiring harness model. For each harness the fitting approach is indicated, full-auto for cases where no manual work was required; semi-auto when manual adjustments were made after automatic fitting; and manual for cases where the automatic alignment method was not used. The tests harnesses are sorted by number of connectors.

Formboard design time reduction  The KBE formboard design applications allow an engineer to create manufacturing drawings in minutes instead of hours. Unfortunately no time data of the individual formboard generation steps was made available. A survey at Fokker Elmo [13] during the initial Knowledge Acquisition (KA) provides indications for the time required to generate a formboard (figure 1.4). Simple harnesses take about 10-20 hours, medium and complex harnesses from 20 hours to over a week. The average time required to generate a formboard drawing from an input PDF for the case presented is 18 minutes, half of which are activities requiring human interaction with the GUI. For larger wiring harnesses such as those presented in section 8.1, generating a drawing can take more than an hour (73 minutes for harness W, of which 40 minutes manual work). In these cases the automatic alignment function does not provide a satisfactory result and manual adjustments are required. Test harness Y took the longest to fit manually: 50 minutes. This is much less than the tens of hours required in the current formboard generation process.
8. Results

8.3 Discussion: industrial potential

The performance indicators for the formboard design process are quality and lead time. This section discusses whether and how the KBE applications to support the formboard design process achieve performance improvements. The section concludes with an quantitative estimation of the achievable productivity increase and the present interest in industrializing the KBE solutions.

The central aim of formboard design is enabling the production of a wiring harness, defined by its 3D design definition, on a 2D manufacturing tool, the formboard. For a high quality formboard, the following is required:

- Formboard frame dimensions and wiring harness layout that enables efficient manufacturing in terms of floor space and manufacturing ergonomics, in a flat plane.

- The transformation of the 3D design definition state to the 2D manufacturing state in a frame, such that the physical wiring harness, manufactured using the 2D tool, will fit in the (3D) aircraft.

- A correct and complete definition of the wiring harness 3D design state.

Meeting these quality requirements is currently a time-consuming process, with specific complexity aspects. These are addressed in separate stages of the formboard design process for each of which a KBE application was developed (see section 4.4). The approach to limit the complexity of the KBE application development process was discussed in section 3.5 and section 4.7 discusses limiting the complexity of the individual applications. The manners in which the KBE applications mitigate or reduce complexity aspects [8] of the formboard design process for a formboard engineer, thereby decreasing lead time and ensuring quality, are discussed next.

Correct and complete design definition Obtaining a correct and complete definition of the 3D design state can be difficult. This mainly due to the nested complexity of Electrical Wiring Interconnection System (EWIS) development process interactions with the formboard design. Different customers use different Computer Aided Design (CAD) systems and adhere (or not) to different design standards, leading to structural complexity of the input models. A lot of time is spent in converting these models or developing algorithms to do so. Assessing the completeness and
correctness of an input model can be difficult. This behavioral complexity makes it necessary in some cases to re-create a design completely.

To make the formboard design applications independent of the various 3D design definition formats (i.e. reduce the structural complexity of interactions between them), a standard input format was defined. The translation of the 3D design definition to this standard is implemented as a separate stage (see section 4.4), in the \textit{f3input} application. This application reparameterizes an input model to the defined standard format. The behavioral complexity of assessing the validity of the reparameterized model is reduced by giving an engineer the opportunity to directly compare input and parameterized components. This is done with High Level Primitive (HLP)s specific for the \textit{f3input} application, that visualize both the input 3D definition geometry and the interpreted parameterization in the same window (or report, see figure 7.8). In fact, the application does not aim at providing a perfect parameterization fully automatically (this would need a more structurally complex application, requiring more development time). Where components cannot be reliably identified, simple directions are asked from the engineer, who can change parameters such as a component centerline axis or the type of a component. This was necessary for a few components in the tested wiring harnesses. In this manner most models can be converted and verified in reasonable time: 1 to 12 minutes per model for the test cases presented in this chapter.

\textbf{Transformation between geometry states} When creating a formboard, engineers spend a large amount of time checking whether a wiring harness can be transformed from its 3D design definition state to a flat state enabling 2D manufacturing on a formboard. This is structurally complex as the states are represented in different models, consisting of many components, where engineers must identify the interrelations within and across models. Switching from and to models, navigating through them and consulting other sources (e.g. handbooks) becomes a time-consuming activity. While comparing the two geometry states, an engineer must determine if flexibility limits are violated or not. The prediction of wiring harness mechanical properties is behaviorally complex and can also present evaluative complexity where different assessment sources or tacit knowledge are used (e.g. expert opinion, handbooks). These complexities are amplified by the behavioral complexity of the automatic flattening method used by the CAD system as the quality of the resulting 2D state is difficult to predict, which leads to additional checks and adjustments.

The \textit{f3flat} application aims at eliminating or mitigating the complexities of the transformation between geometry states for a formboard engineer. The structural
complexity is reduced by combining the different geometric states and their interrelations in a single HLP. Analysis results are attributes of the multi-state HLP and can be inspected by the engineer, to whom flexibility violations are pointed out without having to search for them. This limits the evaluative complexity for the formboard engineer, who can decide to accept or ignore potential problems based on consistent results. The behavioral complexity of predicting bundle flexibility is transferred from the individual formboard engineer to experts managing the flexibility parameters knowledge base. The complexities previously caused by the automatic CAD system flattening method are eliminated by using the more reliable, predictable flattening procedure presented in section 5.5. The need for performing analyses after or during fitting (presenting structural and behavioral complexity) is eliminated by discretizing the $f3flat$ bundle HLPs in bendable segments that are analyzed to provide bend angle constraints. These are provided as constraints on the Degree(s) Of Freedom (DOF) of the $f3fit$ bundle HLPs, ensuring that the the flat, fitted state on the formboard respects the 3D design definition state. The $f3flat$ application ensures all components are analyzed with consistent quality. The tests presented in this chapter show that it enables an engineer to generate and inspect a flattened wiring harness in 1 to 15 minutes.

**Efficient manufacturing**  Transforming the flattened wiring harness state to a state that is within a formboard frame is considered to be the most creative activity in the formboard design process. It is a time-consuming trial-and-error process for an engineer to find a ’good’ configuration amongst the infinitely large number of possible configurations and the interactions between elements in them. Additional to this structural complexity, determining whether a configuration is ’good’ is a case of evaluative complexity. It can be difficult to decide between the different aspects for efficient manufacturing such as frame dimensions and manufacturing ergonomics. Adding manufacturing symbols to the formboard drawing can be especially time-consuming, but is not really a complex task.

The $f3fit$ application GUI removes some of the structural complexity of manual fitting by constraining the engineer to fixed bend segments and flip switches. The automatic alignment method fits most branches, leaving some complex cases for the engineer to solve manually. The evaluative complexity of deciding between formboard frame size and ergonomic layout considerations is reduced by dividing the fitting process in two steps. First the tightest fit without geometry crossings is identified and the smallest corresponding formboard frame size is selected. Within the selected frame a better layout can be searched for, based on a quantitative fitness score, by manual modifications or automatic functions. Standard dress-up symbols are associated with each HLP variant of the $f3fit$ application and are gen-
8.3. Discussion: industrial potential

Discussion

erated automatically. Using the f3fit application, the time required to fit is reduced. Standard dress-up symbols are generated nearly instantaneously. The lead time to perform fitting ranges from 2 to 75 minutes for the tests presented in this chapter.

This chapter has given indications of the time required to generate a formboard, supported by KBE applications. The following illustrates the productivity increase that the reduction in lead time can provide.

Productivity potential  Lacking detailed data on formboard creation time (this is confidential information), a generic -conservative- case can be described to evaluate quantitatively the benefits of automating the process for a company. Assume that the time required to generate a formboard drawing for simple, medium and complex harnesses is 15, 30 and 50 hours, respectively. A conservative estimate of the time required using the KBE applications would be 30 minutes for simple harnesses, 2 hours for medium and 4 hours for complex cases. Add 3 hours to account for pre- and post-processing work that is not automated such as administration, downloading and plotting. If the ratio between simple, medium and complex harnesses is 8:1:1 for a shipset, the average time required per harness is 20 hours in the original case and 4 hours in the KBE-supported case: a five-fold productivity increase.

Industrialization  The potential of the demonstrator applications is recognized by Fokker Elmo. Both the technical experts and management agree that these application could considerably reduce formboard creation costs and time [142]. The Delft University start-up company KE-works, specializing in the development of KBE applications, judges that the demonstrator applications have reached a Technology Readiness Level (TRL) of 5 [143]. Fokker Elmo is seriously considering to raise the TRL and industrialize the f3 demonstrator applications. KE-works has been contracted by Fokker Elmo to write a proposal for this development [143]. At the moment of writing, this proposal has been submitted and is being evaluated by Fokker Elmo. Industrial implementation and application of the developed solutions in wiring production will provide a definitive validation.
Chapter Nine

Conclusions

Process time and product quality of formboard design can significantly be improved by the application of Knowledge Based Engineering (KBE) techniques. The Computer Aided Design (CAD) package nowadays used for generating formboard drawings does not incorporate customer-specific wiring harness manufacturing design requirements. As a consequence it makes the design of a quality formboard drawing a time-consuming and repetitive activity which is mostly manual. A KBE application can capture customer-specific requirements and largely automate design activities, increasing the productivity of formboard engineers. This chapter presents conclusions that confirm the central proposition of this dissertation:

Research proposition. The performance of the formboard design process can be increased in terms of time and quality by largely automating it using Knowledge Based Engineering.

The validity of the proposition has been demonstrated by developing a suite of KBE applications that largely automate formboard design based upon customer-specific requirements and testing these applications on industrial wiring harness designs. The time spent on formboard design tasks was demonstrated to reduce from hours to minutes. A conservative calculation indicates that, in practice, a five-fold productivity increase can be obtained. The quality of the output formboard drawings was approved by formboard design experts. No quantitative measure for an improvement in quality could be defined herein as this required comparisons on the manufacturing floor. It is however reasonable to expect quality improvements as the automated methods perform checks completely, consistently and provide uniformity in the output drawing layouts.

This main conclusion could be drawn by finding answers to the research questions:
Research question 1. How to automate the formboard process steps?

Research question 2. How to model different geometric states of a product and their interdependencies in a KBE application?

Research question 3. What methodology can be used to develop a KBE solution that matches customer-specific requirements?

The following paragraphs provide conclusions and recommendations regarding these questions. In reverse order, starting with question 3 on the development methodology.

KBE development methodology A development methodology for developing design automation solutions based on KBE and search methods was proposed and successfully applied to the formboard design case. The main distinctions with respect to existing methods are the use of software prototypes for knowledge verification/validation and the adoption of an incremental, agile development approach. Knowledge Engineering (KE) methods such as Methodology and software tools Oriented to Knowledge based engineering Applications (MOKA) focus mainly on knowledge acquisition and structuring, while Software Engineering (SWE) mainly is about development of the software itself. The proposed approach combines methods from the fields of Systems Engineering (SE), KE and SWE. Iteratively, key processes and requirements are identified; knowledge is acquired, formalized, implemented in a KBE application and verified with domain experts, see figure 9.1. New knowledge will inevitably emerge during a design process, prototype applications provide an additional means for elicitation of knowledge. They also allow evaluation of the development product (the application) instead of a half-product (knowledge repositories).

The development methodology should be applied to other use cases and improved based on these experiences. It is doubtful that a single recipe for KBE application design will emerge, as design cases and industrial or academic environments differ. Instead a set of lean KBE development principles could be agreed upon by KBE practitioners.

KBE building blocks KBE applications can be constructed using flexible parametric building blocks, notably High Level Primitive (HLP), Capability Module (CM) and Structuring and Collating Node (SCN) classes. HLPs and CMs were proposed by La Rocca [17]: HLPs incorporate declarative and procedural knowledge to generate
Figure 9.1: KBE application development approach.

Decomposing a design process in stages, and developing a KBE application for each stage has certain pros and cons with respect to the complexity of the developed solution. To find a balance a Multiple Domain Matrix (MDM) is proposed to relate design process activities to HLPs, providing a tool to identify design stages and define stage-specific HLPs. The formboard design process was divided in three stages, for which applications were developed: \textit{f3input}, \textit{f3flat} and \textit{f3fit}.

HLPs with multiple geometric representations, or states, are introduced. In many design applications the focus is on a single geometry, but this need not always be the case. In manufacturing design for example the design and manufacturing geometries of the same product can be different. HLPs can be defined with geometry attributes corresponding to different states, which share parameters and may have interdependencies. The \textit{f3flat} HLPs feature multiple geometry states, each component has a 3D and a 2D representation as shown in figure 9.2.
This work adds a third type of KBE building block: the SCN class aggregates HLPs and other SCNs and defines interdependencies between these. While a HLP is a stand-alone class that can be reused in different product models structures, a SCN contains rules defining the product model structure. Where in previous work CMs were mainly defined to act on a developed product model, this work presents cases where product model rules depend on CM analysis results.

**Formboard automation approach** Automation strategies have been developed for each step in the formboard design process. This requires more than simple emulation of human activities as these are often constrained by CAD package capabilities and based on tacit judgments: new methods are developed and existing approaches are formalized.

Procedures to analyze the 3D state of a wiring harness for flexibility violations...
in bundles and 3D break-out shapes at connection points have been developed. To perform flexibility evaluations a proprietary empirical method, validated by years of application at Fokker Elmo, was selected. A new method for wiring harness flattening was devised and implemented that flattened states of the wiring harness bundle and connection-point HLPs individually, based on flexibility analysis results and aligns them in a 2D plane by applying twist to the bundle HLPs. The approach returns models where: the flat connection point geometry closely matches the 3D shape, bundles are straightened and flattened only when allowed, and only the minimum amount of twist is introduced in bundle sections.

The flattened model is discretized into bendable sections for which bending limits are established. Both manual fitting by an engineer and automatic fitting using search methods are constrained by these limits. From experimentations with different fitting approaches the so-called alignment method, based on manual fitting heuristics, was selected and implemented. This method attempts to align branches of bundles with respect to each other according to a limited set of bending and flipping strategies. The ergonomics of wiring harness manufacturing was analyzed providing rules for formboard layout and symbols used.

The exchange of 3D CAD data is a known difficulty in multi-disciplinary, concurrent engineering environments involving multiple suppliers. This work does not attempt to solve the interoperability issue directly, but proposes and demonstrates using neutral standards available in CAD systems such as STEP AP214. The static geometry model provided by the STEP standard can then be transformed into a HLP parameterization by a 'interpreter' KBE application. Such an interpreter application was successfully developed to import 3D wiring harness models designed with CATIA V5.

**Recommendations** This research has focused on solving formboard generation for the vast majority of wiring harnesses. At the time of writing, new research initiatives have been initiated to develop solutions for some special cases: closed loop wiring harnesses, which occur more frequently as the number of harness separations increases; and semi-3D formboards, which allow wiring harnesses that cannot be flattened to be manufactured. Although the latter case does not occur frequently, creating such a semi-3D formboard is very time-consuming\(^1\). Apart from generating formboard drawings, engineering time is spent on modifying existing ones due to 3D definition changes as well. Approaches to support the identification

\(^1\)Work on these subjects has and is done by Master students. Preliminary research into flattening closed-loop wiring harnesses was performed by Jelle Veraa’s [144]. Automatic semi-3D flattening and 3D block design are presently being developed by Pieter van Assche [145].
of these changes and selectively adapt existing formboard configurations could be developed. The KBE applications have been tested using 3D models from a select number of aircraft projects. To improve the applications and evaluate their performance, further tests on a larger number of harnesses from different projects should be performed. The flexibility analyses use an empirical method based upon experimental data and are limited by the availability of this data. Further research should be performed to develop and validate higher fidelity models. The alignment method as presented is the most successful of the automatic fitting approaches that have been experimented with so far. The method itself can be improved by addition of different bending and fitting strategies, and by implementing different search methods. Other fitting and formboard configuration optimization approaches can be investigated as well. More specific recommendations can be found in the chapter discussion sections.

**Final remarks**  The developed formboard KBE suite offers functionalities that are not present in conventional CAD systems. Given that very specific knowledge is required for these functionalities, it is generally not in the interest of the CAD company to invest in implementing such functionalities in their general-purpose CAD system. Neither is it in the interest of Fokker Elmo or similar companies to give their specific knowledge away, making it available to competitors. The developed KBE applications can considerably reduce the amount of repetitive work, while ensuring compliance to physical constraints and manufacturing guidelines. It is recommended to raise the Technology Readiness Level (TRL) of these tools in order to apply them in industrial practice at Fokker Elmo. Design automation by KBE and search has the potential to provide a large contribution in increasing engineering productivity, which is needed for Europe’s aviation industry to remain competitive.
Bibliography


[51] Knowledge Transfer Partnerships. The case for fans at Rolls-Royce.


[120] University of Konstanz. graphdrawing.org, 2013.


[160] H. Mooz and K. Forsberg. A Visual Explanation of Development Methods and Strategies including the Waterfall, Spiral, Vee, Vee+, and Vee++ Models. In *Proceedings of the international council for systems engineering (INCOSE) conference*, number Figure 1, Melbourne, Australia, 2001. INCOSE.


Appendix A

Characteristics of a KBE language

This appendix describes the main characteristics of a KBE system, using General-purpose Declarative Language (GDL) from Genworks [53][54] as an example. Like most KBE systems, GDL is a superset of Lisp, making available the complete Lisp language when coding. GDL specifically is based on ANSI Common Lisp, and makes use of Common Lisp Object System (CLOS) [146]. Now follows a summary of the overview on KBE language characteristics from La Rocca [18] and Cooper [34].

A KBE language is a programming language supporting the Object Oriented (OO) paradigm that allows operators to define classes, superclasses and object hierarchies, including relations of inheritance, aggregation and association. In GDL the define-object operator defines a class, which can inherit from other classes via a mixin list. A define-object class contains input-slots and computed-slots, both representing property-value pairs to be passed-in or containing expressions to be evaluated. It also contains objects, aggregate classes to be instantiated as child objects, having a class and input slot specification. Regular Lisp functions can be used, but class-specific functions can be defined for a define-object as well (methods in OO programming). Associations between slots are set by reference chains, that are preceded in GDL by the ‘the’ operator. An example of a define-object including a reference chain is given in appendix G. A combination of define-object classes is a product model, that can instantiate an object tree.

Lisp based KBE languages feature dynamic typing, which means that values have types, but attributes (slots) do not. This enables modification of the product tree topology at runtime. The order in which objects and slots are declared is not relevant, therefore the coding style is declarative. Lisp actually allows the use of a procedural coding style as well, and is therefore called a multi-paradigm language. A language used for Knowledge Based Engineering (KBE) would also require runtime caching and dependency tracking. The first allows memorizing results of computed values at runtime and the second allows the system to keep track of the validity of cached values, enabling associative modeling. As caching can lead to high mem-
ory usage, a garbage collector performs *automatic memory management*. Another characteristic of a KBE system is that it can use both backward and forward chaining inference mechanisms. For a typical KBE application backward chaining, or *demand driven evaluation*, is preferred as it prevents waste of computational resources: expressions are only evaluated to satisfy a direct user request. [18]
Search techniques

There are various strategies to explore the design space. For a simple problem with a few parameters it may be possible to compute all combinations and select the best one. Such an exhaustive search is generally not affordable when the search space becomes impossibly large due to the combinatorial explosion or when the evaluation of the objective and/or constraint function is very computationally expensive. A search tree or graph can be explored using uninformed strategies such as breadth-first and depth-first, or informed -heuristic- strategies, such as best-first and A* search.

The most basic local search strategy is the greedy search, where the algorithm always selects the best neighbouring state. Such a method has a high risk of getting stuck in a local optimum however. Metaheuristic methods allow escaping a local optimum. Simulated annealing is such an algorithm, where instead of always picking the best neighbour, a successor can randomly be selected. The probability that this happens will decrease progressively, analogous to a metallurgy annealing cooling schedule. A local beam search evaluates $n$ states and selects the best successors of these states. This becomes a stochastic local beam search when successors are selected randomly. Genetic algorithms are a stochastic local beam search variant where successor states are created by combining the fittest parent states (crossover) and randomly modifying states according to a certain probability (mutation). Other metaheuristics are for example ant colony optimization, tabu search and particle swarm optimization. Heuristic methods can be and are often tailored to a specific problem, based on domain specific knowledge on how to find a design solution. These are often variants of the metaheuristic search methods. Note that all the abovementioned search methods do not guarantee that an optimum solution will be found, but provide a way to obtain feasible solutions [60].

In design spaces that are continuous differentiable, gradient based optimization methods can be used. A gradient based method determines a search direction, performs a one-dimensional search, tests for convergence and if false repeats this procedure [147]. Many engineering design problems feature discrete variables, non-
smooth functions or a non-convex design space and therefore do not allow the use of gradient methods. Often different optimization methods are combined for both local and global optimization.

**Multiple objectives** Problems with multiple objectives that cannot all be optimal simultaneously are so-called Multi Objective Optimization (MOO) problems [148]. A design that cannot be improved with respect to a certain objective without worsening at least one of the other objectives is a Pareto optimal design, and a set of Pareto optimal designs forms a Pareto front. To find a final design, a human operator can select a design point on the Pareto front; or the objectives can be evaluated in order according to their importance; or a goal, or aspiration levels, can be set for the objectives (goal programming). A MOO problem can also be reduced to a Single Objective Optimization (SOO) problem by combining the objective functions into a single one using weighting factors (compromise programming) or turning all but one objective into constraints.

**Multiple disciplines** Most engineering design will involve different disciplines with couplings between the disciplinary aspects. In traditional design disciplines are often considered separately leading to the situation where couplings make design more difficult and are not exploited for synergetic effects [148]. According to Kroo [149] the goal of Multidisciplinary Design Optimization (MDO) is to "to provide a more formalized method for complex system design than is found in traditional design approaches". MDO provides approaches to formalize a multidisciplinary design problem such that search/optimization techniques can be applied to explore the design space. Formalizing the multidisciplinary design problem demands careful system analysis and decomposition (by object or aspect) of the problem in smaller problems. Coordination is required to manage the couplings between subsystems during optimization.

**Design and Engineering Engine** A MDO framework is being developed at the Flight Performance and Propulsion (FPP) group of Delft University of Technology to support the multidisciplinary design of aircraft: the so-called Design and Engineering Engine (DEE) with at its core the Multi Model Generator (MMG) [151]. The MMG is a KBE application that generates a product model design and disciplinespecific models or views for analysis. The DEE structure is depicted in figure B.1 with the MMG at the right side. It consists of an initiator that initializes the parameters for the MMG based on a set of requirements; the MMG; disciple analysis tools; and a search toolbox checking for convergence and compliance to requirements. Example implementations of the DEE concept can be found in [152] and [153].
Figure B.1: The Design and Engineering Engine [150]
Appendix C

**Elements of Systems Engineering**

Today, Systems Engineering is an essential methodology for developing complex systems. Van Hinte and van Tooren describe it as "a management template for complex engineering projects" [81]. Today Systems Engineering (SE)'s scope encompasses the entire life cycle of a system [40]. Literature shows there are many definitions of SE, such as the definition of the International Council on Systems Engineering (INCOSE) [154]:

"... an interdisciplinary approach and means to enable the realization of successful systems. It focuses on defining customer needs and required functionality early in the development cycle, documenting requirements, then proceeding with design synthesis and system validation [...]."

Van Hinte and van Tooren [81] point out that SE is not a recipe for good design. Instead, SE can be regarded as an approach that provides means to develop a good design. These means are formal methods for requirements analysis, functional analysis, design synthesis and verification [40]. These are the fundamental elements of the systems engineering process, depicted in figure C.1. System analysis and control techniques are used to manage these elements including aspects like schedule, performance and risk.

During the requirements analysis process inputs, including customer requirements, are analyzed to develop functional and performance requirements and identify constraints. These can be structured in a requirements discovery tree. Figure 2.14 illustrates functional requirements and constraints for the formboard process, derived from the Electrical Wiring Interconnection System (EWIS) development process. Functional flow diagrams and N2-charts are SE tools that can be used for functional analysis and allocation. A N2-chart (or a Design Structure Matrix (DSM)) allow showing interdependencies between elements of a decomposed system (e.g. functions, solutions). During design synthesis concept solutions are
developed and trade-off are made. The requirements loop and the design loop reflect the iterativeness of the design process as functions emerge from design concepts, and requirements emerge from the functional analysis. For verification, the designed solution is compared to the requirements at different system levels, often structured in the form of a V-model as illustrated in figure C.2.

These SE methods are the backbone of developing a complex engineering system such as KBE design automation solutions.
Figure C.2: Systems Engineering and Verification [40]
Elements of Knowledge Engineering

Effective use of knowledge can increase the productivity of a knowledge worker, leading to the need for proper Knowledge Management (KM). Drucker [155] considers knowledge workers to be a key asset of a 21st century organization. Knowledge Engineering (KE) is a scientific methodology to analyze and engineer knowledge [37], it is a subset of KM that focuses on the capture of knowledge in computer systems.

The definition of knowledge is the subject of a long-lasting philosophical debate without any consensus being reached as yet [44]. A distinction is often made between data, information and knowledge [37]. Plain data is uninterpreted, e.g. characters; information is data with a certain meaning, e.g. characters combined into words; knowledge is interpreted/internalized information that people can practically use to carry out some task and create new information, e.g. a recipe. These distinctions depend on context however, as Schreiber et al. [37] argue, "one person's knowledge is another person's data", consider for example a recipe in a foreign language.

Knowledge can be considered to have different dimensions [156]: explicit vs. tacit and conceptual vs. procedural. Explicit knowledge is readily available in one's mind, e.g. how to solve an equation. Tacit knowledge is knowledge that one is not explicitly aware of, that cannot be formalized and is deeply compiled in one's mind, e.g. how to find the keys on a keyboard, or how to cycle. Conceptual and procedural knowledge correspond to knowledge about concepts and activities, respectively. These dimensions are not black and white, but lie on a scale. Consider a person learning to drive, initially knowledge on how to drive will be mainly explicit, but this will shift to tacit as the driver becomes more experienced (this is called internalization of knowledge).

The KE process aims at producing a knowledge repository (Knowledge Base (KB), Knowledge Based System (KBS) or Knowledge Based Engineering (KBE) system) and is characterized by three main steps: the acquisition of knowledge, its
D. ELEMENTS OF KNOWLEDGE ENGINEERING

implementation in a software system and verification.

Knowledge Acquisition Milton [156] defines Knowledge Acquisition (KA) as the capture, analysis and modeling of knowledge. Capturing knowledge can be quite an effort requiring different techniques. These include capturing both explicit and tacit knowledge from domain experts, retrieving knowledge from documentation (handbooks, textbooks) and observation of the engineering process and products. Acquired knowledge can be categorized in three levels of detail [156]:

- Basic knowledge: essential advice, task outline, typical situations and issues
- Detailed knowledge: full description of major tasks, situations and issues
- Very detailed: comprehensive description, thorough consideration of situations and issues.

The effort required to acquire higher levels of knowledge increases linearly according to Milton [156]. More resources and time are therefore required in order to reach the very detailed level of knowledge necessary to automate a design process. This can be an issue in an industrial environment where domain experts fulfill critical roles in the business process and only limited time can be spent on involvement in a KE process.

Captured knowledge will often be incomplete, contradictory and possibly biased. Different experts will provide different descriptions of the same process and product. Typically the views on a process given by experts will be descriptive (how it is done), handbooks on the other hand are often prescriptive accounts (how it should be done). Differences are to be expected, as designers often do not follow prescriptive models exactly [61] and these models are often not entirely up-to-date. To be of practical use and to identify discrepancies between expert views, observations and other material, captured knowledge is analyzed and structured into models. If automation is the objective of the KE process, prescriptive models are required. Two approaches describing techniques to perform KA are the 47-step method by Milton [156] and the CommonKADS method [37].

These KA approaches provides strategies to collect knowledge and represent it in models. In early phases informal modeling techniques are applied which are gradually developed into more formal models as the KA progresses. Some techniques to elicit knowledge from experts are interviews, process mapping, teach back, etc (see [156]). Methods to formalize the acquired knowledge have been suggested, such as
the 'k-page' [156] and the ICARE forms [36]. This last acronym stands for the different elements of the MOKA informal model: Illustrations, Constraints, Activities, Rules and Entities (see figure D.1). The other approaches feature similar elements that are organized using different diagrams such as: product and process concept trees or maps showing composition, hierarchy, attributes, relations; matrices showing dependencies, relations; state diagrams; frames or annotation pages. These more informal models are developed into formal models, using Unified Modeling Language (UML) based notations, mainly activity diagrams and class or object diagrams. The MOKA methodology for example provides the UML derivative MOKA Modeling Language (MML) [157]. MOKA organizes the formal model into a product model (MML class diagrams providing structure, function, behavior, technology and representation views) and a design process model (MML activity diagrams and strategy tables). The product model roughly corresponds to the CommonKADS domain layer and the design process model to the other CommonKADS tasks, inference and strategy layers.

Figure D.1: MOKA methodology features. Left: Informal knowledge is modeled with ICARE forms and diagrams. Right: Formal knowledge is represented in a product model (MML class diagram) and a design process model (MML activity diagram). From [63]

Modeling knowledge in knowledge bases during KA is supported by dedicated software tools as PCPACK [69] and Protégé [158] for instance. PCPACK actually includes the ontologies of the MOKA informal and formal models and was used to model the formboard process knowledge as will be discussed in section 3.4.

Knowledge implementation  KE provides methods for the implementation of knowledge in KBS: computer applications used to store knowledge for solving problems
in a specific domain [17]. KBE applications are a special type of KBS, as was described in section 3.1.1. Although the focus here is on implementation of knowledge in automation software, KA can be used as a teaching method or to set up other knowledge repositories such as handbooks, manuals, databases, etc. The PC-PACK [69] program for example enables the user to create a 'knowledge web' from a knowledge base. The MOKA project [159] presents some preliminary work on automating the translation of a formal knowledge base to a KBE application.

Knowledge verification Verification and validation of the knowledge acquired, modeled and implemented is essential for the success of a KE effort. This is typically performed by teach-back and showing different views of the acquired knowledge base to domain experts, e.g. ICARE forms, diagrams, relationship matrices. The KE methodologies usually require thorough validation of a knowledge base before proceeding to the implementation in a KBS or KBE application. A more fruitful approach to elicit expert knowledge and general feedback is not only to show the knowledge base but also the use case of the software tool, which implements that knowledge. This elicits different reactions from domain experts and hence complements the 'model-based' knowledge verification activities. Demonstrations of working prototype application typically elicit expert reactions such as: "can the program also do...?" and "did you think of...?". Knowledge and its implementation are ultimately validated when experts agree that the behavior of the engineering software tool is as desired, i.e. matches the expected product or process behavior.

Product development processes use Systems Engineering (SE) to manage product quality. Developing a KBE application to support the product development process requires the application of KE to the SE process. SE tools such as functional flow diagrams, design option trees, are used for KA. The SE requirements and functional analysis phases could be considered a type of KA phase. The SE process synthesis phase and verification correspond to the KE implementation and verification steps, respectively.
Appendix E

Elements of Software Engineering

Systems Engineering applied to the development of software yields Software Engineering (SWE). This section provides an overview of different software development approaches, starting with traditional, plan-driven methods and continuing with agile approaches.

Some of the most applied traditional development approaches are [160]: The waterfall model (Royce, 1969) consists of sequential steps of requirements definition, design, implementation, verification and maintenance. The spiral model (Boehm, 1980) aims at resolving risk in iterative phases before proceeding with the waterfall approach. The ’V’ model (NASA, 1987) is derived from the waterfall and maps test phases to development phases.

To increase development flexibility, and perhaps because developers were not following a prescribed methodology, agile software development methods have been introduced. Agile methods such as Extreme Programming, Scrum and Feature Driven Development emerged as an alternative to the traditional development approaches that were felt to be heavyweight and documentation-driven [161]. A software development approach is said to be agile when it is [162]:

- Incremental (small releases, rapid cycles)
- Cooperative (customer and developer in close communication)
- Straightforward (comprehensible method, well-documented)
- Adaptive (able to make last moment changes)

The development of agile methods reflect a desire for a more lean approach to software engineering in the same way lean principles have been successful in manufacturing and development. Poppendieck [163] presents lean principles for the
software development process that can be translated into agile practices. To summarize these principles: (1) eliminate anything that does not add value to the product (i.e. waste); (2) development is a discovery process, amplify learning; (3) deliver as fast as possible to obtain reliable feedback and enable (4) deciding as later as possible; (5) empower the team; (6) build integrity in; and (7) see the whole.

The ’Agile Manifesto’ [161] distinguishes agile from traditional approaches in terms of emphasis on: "individual and interactions over processes and tools; working software over comprehensive documentation; customer collaboration over contract negotiation; and responding to change over following a plan". This last point is reflected by Boehm [164], who characterizes the traditional software development methods as more plan-driven. Each method can be placed in a scale of emphasis on plans (and documentation). According to Boehm, the suitability of either agile or plan-driven methods depends on the specific case and combined approaches are possible. Typical properties of agile and plan-driven development cases are [164]:

- **Agile**: rapid value as the objective, small development teams, knowledgeable and collaborative developers, rapid change and emergent requirements.

- **Plan-driven**: high assurance as the objective, large development teams, adequately skilled developers, largely stable and knowable requirements.

A common misconception is that using an agile method is equivalent to messing around, not keeping to requirements and cannot be controlled. Agile methods allow developers more freedom, but do use formal control mechanisms and do require producing documentation and tests. The more formal control mechanisms are used, the more the methodology shifts towards the plan-driven side of the scale. McConnell [38] argues that developing software is a heuristic process that differs per project and therefore advises using a mixture of methods instead of opting for a specific methodology.

Most Knowledge Based Engineering (KBE) applications will be extensive, complicated software applications. The proper use of SWE design tools (Unified Modeling Language (UML), pseudo-code, etc.) and methods can reduce the complexity of designing software or enable to cope with it. The use of agile development principles reduces the complexity of having to identify all the relationships, requirements, etc. before starting developing. The high level of communication and feedback can reduce the evaluative complexity of the design process.
Agile methods have been proposed for the development of Knowledge Based System (KBS). Knublauch [165] argues that the evolvability of knowledge models and the need for collaboration when developing KBS make Extreme Programming a suitable development method. An agile approach fits indeed with an iterative Knowledge Engineering process which is expected to have largely emerging requirements and solutions. As Knowledge Engineering (KE) is typically performed by small teams, an approach based on agile principles can be adopted to develop KBE modules. In order to ensure control of the development process, plan-driven elements are required as well.
Appendix F

**Existing KBE development methodologies**

The KBE development methods available are very similar to each other and all feature the Knowledge Engineering (KE) phases discussed in appendix D.

A KE process from a business model perspective is proposed by van der Elst and van Tooren [27], as shown in figure F.1. The approach consists of six phases:

1. Process analysis, to identify opportunities for KBE application
2. Knowledge Acquisition, to capture and validate expert knowledge
3. Knowledge Structuring, to redesign the process and determine automation strategies
4. Knowledge Application, to develop KBE modules
5. Integration and deployment, to combine the KBE modules in a Design and Engineering Engine (DEE)
6. Business implementation, for support, maintenance and training.

The boxes in figure F.1 represent deliverables. Van der Elst and van Tooren [27] specify that these process steps are iterative and continuous collaboration between developers and domain experts is necessary.

The Knowledge Based Engineering (KBE) life cycle, illustrated in figure F.2 was one of the main results from the Methodology and software tools Oriented to Knowledge based engineering Applications (MOKA) project [36] [63]. It starts by identifying and justifying an opportunity for KBE, followed by a knowledge capture and formalization phase, the application development phase or 'packaging' phase and finally the deployment phase, after which a new cycle may be started. The MOKA methodology focuses on the capture and formalize steps (see appendix D on Knowledge Acquisition (KA)). A large amount of informal knowledge must be
collected and verified to create and validate formal models, from which knowledge is implemented in a KBE application. No specific methods are provided for the development of software itself. The MOKA methodology is probably the most cited approach to develop KBE applications in literature, for example in [46] [75]; and is often the starting point for methodological research. A useful example is the work by Chan [87], who extends the MOKA methodology with activities at different abstraction levels (problems, disciplines and tools, see figure F.3) in order to model simulation work flows that can be used to develop Multidisciplinary Design Optimization (MDO) frameworks.

The two development methodologies above were envisioned specifically for developing a KBE application, the CommonKADS methodology [37] however proposes a ‘configurable life cycle’ model that focuses on the management of knowledge systems projects. This model is illustrated in figure F.4 and is based on the spiral model from Software Engineering (SWE). It consists of four quadrants indicating project management stages: review, risk, plan and monitor [37]. The actual development work takes place during the ‘monitor’ phase.

Other methodologies are described for example by Lovett [167], Curran [168] and van der Velden [62].
Figure F.2: The MOKA methodology KBE life cycle. From [17], based on [166]
**F. Existing KBE development methodologies**

<table>
<thead>
<tr>
<th>Actor</th>
<th>Step</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Design engineer</td>
<td>Step 1: Define problem statement</td>
<td>Initial ICARE PDT</td>
</tr>
<tr>
<td>Design engineer</td>
<td>Step 2: Structure the problem and disciplines</td>
<td>N² diagram</td>
</tr>
<tr>
<td>IT engineer</td>
<td>Step 3: Model the activities</td>
<td>BPMN workflow diagram</td>
</tr>
<tr>
<td>Knowledge engineer</td>
<td>Step 4: Fill in ICARE PDT-forms</td>
<td>formal ontology</td>
</tr>
<tr>
<td>Knowledge engineer &amp; IT engineer</td>
<td>Step 5: Formalise process knowledge</td>
<td></td>
</tr>
</tbody>
</table>

Figure F.3: Extended MOKA methodology for modeling simulation workflows. From [87].

Figure F.4: The CommonKADS configurable life cycle model for managing knowledge systems projects (right), based on the spiral model (left) [37]
Appendix G

KBE UML class diagrams

This section presents specific aspects of the Unified Modeling Language (UML) class diagram notation used throughout this work. The notation is specifically for KBE languages such as GDL. It emerged as the most convenient style by trying different variants in practice and discussing their advantages and disadvantages. The contributors to these discussions and the variant selection are Durk Steenhuizen, Reinier van Dijk, Gianfranco La Rocca and the author. This section will only address UML features that are of particular importance in the context of a GDL-based KBE application. For more general information on UML, see for example [169] [170].

Class Figure G.1 shows a wing class, both in UML and in GDL code. The class name corresponds to the define-object name. A stereotype can be used to classify the class, for example to indicate whether it is a High Level Primitive (HLP), Capability Module (CM) or Structuring and Collating Node (SCN) class. In a KBE...
class a distinction is made between two types of attributes: input-slots and computed-slots and operations are called functions. To simplify a diagram, a class can be collapsed as illustrated in the bottom of figure G.1. A KBE class can also aggregate a number of child-classes as explained next.

**Composition class diagram variants** In GDL, the objects list contains the subclasses a class is composed of. As shown in the top-left of figure G.2, a subclass or child-class has a class name (e.g. main-wing) and a class type of specialization (e.g. wing). When instantiated, the objects shown in the top-right corner are generated. UML as a language is very flexible, and consequently there are different ways in which the GDL code could be represented in a class diagram, as indicated in the lower part of figure G.2. Deliberations between the persons named before resulted in the selection of the variant with the ‘child-object’ notation. This approach amongst others best allows indicating dependencies between child-classes, varying multiplicities and class specializations depending on rules.
A child-class  Consider an aircraft class which is composed of a main-wing that is a specialization of the wing class. The main-wing child-object is represented as a class linked to its parent by a composition link and the corresponding multiplicity. The child class will be given a type in GDL, which is shown as a specialization link. When input-slots are set in the parent class (e.g. :span 34), these are indicated as attributes of the child-object.

Generalizations - the mixin list  Figure G.4 shows a BWB class that is a specialization of the aircraft class. This means that the BWB class inherits the input-slots, computed-slots, objects and functions from the aircraft class, except if overwritten by the BWB class itself. The generalized classes are specified in the so-called mixin list as indicated in figure G.4. Note that multiple classes can be specified in a mixin list.

Dependencies between classes  The class diagram of figure G.5 shows an aircraft class composed of a main-wing and a winglet child-object, that are each of type wing (a HLP for wing-like components). As indicated in the code, an attribute of winglet (:position) is dependent on an attribute of main-wing (:tip). The existence of a dependency is indicated by the dashed arrow in the class diagram.
The aircraft class is a generalization of the BWB class.

Figure G.4: A KBE UML class that is a generalization of another class (left) and the corresponding GDL code (right).

The winglet child-class is dependent on the main-wing child-class.

Figure G.5: A KBE UML class with two children of the same class type where one depends on the other (left) and the corresponding GDL code (right).
**Conditional specialization**  In many KBE applications rules will determine what type of component to generate. This is illustrated in figure G.6 where the empenage child-object of the aircraft class is either a specialization of the V-tail class or the T-tail class.

![Diagram of conditional specialization](image)

Figure G.6: A KBE UML class where the specialization of the child depends on a condition (left) and the corresponding GDL code (right).
Appendix H

Class diagrams of the f3flat application

The elements of the class diagram of figure 5.24 are further elaborated here:

- connection-point class: figure H.1
- bundle class: figure H.2
- bundle-section class: figure H.3
- endpoint class: figure H.4
Figure H.1: Class diagram of the f3flat application connection-point class.
Figure H.2: Class diagram of the f3flat application bundle class.
Figure H.3: Class diagram of the f3flat application bundle-section class.
Figure H.4: Class diagram of the f3flat application endpoint class.
Appendix I

Input formats

Figure I.1: Schematic overview of the inputs for the f3flat application (excluding bundle- or endpoint-attached components).
I. INPUT FORMATS

```
### START OF THE F3 INPUT FILE
### file created on 5-2-2013 at 14:6.

### WIRE HARNESS INFORMATION
id [wire-harness-id]
assemble [assemble]

### CLOSED LOOPS
closed-loops nil

### BUNDLE LIST
bundle-list {
  \id "Flexible Curve.2 (2)"
    #(24.58204243658543 -9.858521465561878 4.385134275243513)]
  lines are skipped here
  [...] #24.344088242 -9.59498972611 4.30278906785))
  (1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0)
  (0.05851473088593483 0.05851473088593483 0.05851473088593483)

### OPEN LOOPS
open-loops nil

### END OF THE F3 INPUT FILE
```

Figure I.2: Example input file for the F3lat application. The file corresponds to test harness O (figure J.13). Continued in figure I.3.
Figure I.3: Figure I.2, continued.
Figure I.4: Schematic overview of the inputs for the f3fit application.
Figure I.5: Example input file for the f3fit application. The file corresponds to test harness O (figure J.13). Continued in figure I.6.
Figure I.6: Figure I.5, continued.
Appendix J

Formboard drawing results

All figures in this appendix are scaled versions of large scale PDF formboard drawings, generated with the \texttt{f3fit} demonstrator application. They may have been split up in sections to be viewable.

J.1 Company validation test harnesses
Figure J.1: Wiring harness model V at intermediate steps during fitting and dress-up. Screenshots of the f3fit application and the exported PDF.
Figure J.2: Wiring harness model W at intermediate steps during fitting and dress-up. Screenshots of the *f3fit* application and the exported PDF.
Figure J.3: Wiring harness model X at intermediate steps during fitting and dress-up. Screenshots of the f3fit application and the exported PDF.
Figure J.4: Wiring harness model Y at intermediate steps during fitting and dress-up. Screenshots of the f3fit application and the exported PDF.
J. FORMBOARD DRAWING RESULTS

J.2 Application performance tests

Figure J.5: Wiring harness model A, fitted with the f3fit application and exported to PDF.

Figure J.6: Wiring harness model C, fitted with the f3fit application and exported to PDF.
Figure J.7: Wiring harness model F, fitted with the f3fit application and exported to PDF.
Figure J.8: Wiring harness model G, fitted with the *f3fit* application and exported to PDF.

Figure J.9: Wiring harness model H, fitted with the *f3fit* application and exported to PDF.
J.2. Application performance tests

Figure J.10: Wiring harness model K, fitted with the *f3fit* application and exported to PDF.

Figure J.11: Wiring harness model L, fitted with the *f3fit* application and exported to PDF.

Figure J.12: Wiring harness model N, fitted with the *f3fit* application and exported to PDF.

Figure J.13: Wiring harness model O, fitted with the *f3fit* application and exported to PDF.

Figure J.14: Wiring harness model P, fitted with the *f3fit* application and exported to PDF.
Figure J.15: Wiring harness model Q, fitted with the f3fit application and exported to PDF.

Figure J.16: Wiring harness model R, fitted with the f3fit application and exported to PDF.
J.2. Application performance tests

Figure J.17: Wiring harness model S, fitted with the f3fit application and exported to PDF.

Figure J.18: Wiring harness model T, fitted with the f3fit application and exported to PDF.

Figure J.19: Wiring harness model U, fitted with the f3fit application and exported to PDF.
Samenvatting

Knowledge Based Engineering technieken ter ondersteuning van het ontwerpen van productiemiddelen voor vliegtuigbekabeling.

Om de Europese luchtvaart industrie competitief te laten blijven in een wereld met toenemende concurrentie, kortere ontwikkeltijden, complexere producten en minder technisch geschoold personeel is een toename in productiviteit nodig. Fokker Elmo, een Nederlands bedrijf gespecialiseerd in het vervaardigen van vliegtuigkabelbomen, wordt met deze uitdagingen geconfronteerd. Op dit moment is het voorbereiden van de productie van een kabelboom een repetitief, tijdrovend en hoofdzakelijk handmatig, op ervaring gebaseerd proces. Een ingenieur moet ervoor zorgen dat de in 3D ontworpen kabelboom efficiënt gefabriceerd kan worden op een 2D productietafel, het vormbord, zo dat het gefabriceerde product geïnstalleerd kan worden in het 3D vliegtuig. Bestaande Computer Aided Design (CAD) pakketten houden geen rekening met de bijhorende klant specifieke eisen, wat leidt tot veel controles en aanpassingen van een model.

Knowledge Based Engineering (KBE) en zoekmethoden worden voorgesteld om de productiviteit van ingenieurs die vormborden ontwerpen te vergroten en consistentere productkwaliteit te behalen. KBE systemen kunnen product- en proceskennis vangen en systematisch hergebruiken om repetitieve, niet creatieve ontwerp taken te automatiseren. De iteratieve stappen van een ontwerpproces kunnen geautomatiseerd worden door zoek- of optimalisatiemethoden toe te passen. Dit leidt tot de onderzoekstelling van dit werk:

Onderzoekstelling. De prestaties van het vormbord ontwerpproces in termen van tijd en kwaliteit kunnen verhoogd worden door het grotendeels te automatiseren gebruik makend van Knowledge Based Engineering.
De geldigheid van deze stelling wordt getoetst door KBE oplossingen die het vormbord ontwerpproces ondersteunen te ontwikkelen en te evalueren. Om deze oplossingen te kunnen ontwikkelen dienen de volgende onderzoeksvragen te worden beantwoord:

1. *Hoe kunnen de vormbord ontwerpproces stappen geautomatiseerd worden?*

2. *Hoe kunnen de verschillende geometrische toestanden en hun onderlinge afhankelijkheden in een KBE applicatie gemoduleerd worden?*

3. *Wat voor methodologie kan gebruikt worden om een KBE oplossing te ontwikkelen met klant specifieke eisen?*

**Ontwikkelmethode**  

**KBE product model**  
KBE applicaties kunnen gebouwd worden van flexibele parametrische bouwstijlen: ’High Level Primitives’ (HLP) om product componenten te genereren en ‘Capability Modules’ (CM) om aspectmodellen uit de HLPs te extraheren. Het concept van de ’Structuring and Collating Node’ (SCN) wordt in dit werk geïntroduceerd om classes te specificeren die HLPs en CMs aggregeren en onderlinge afhankelijkheden definiëren. In plaats van een enkele KBE oplossing te ontwikkelen, kan het voordelen opleveren om het ontwerpproces in verschillende fasen op te delen. Een ’Multi Domain Matrix’ (MDM) kan gebruikt worden om verschillende stappen te identificeren. Dit leidt tot applicaties gebouwd met fase-specifieke HLPs. De productiemethode van kabelbomen vereist dat de definitie van de kabelboom wordt getransformeerd van zijn 3D ontwerp toestand naar een 2D productietoestand. Om dit te modeleren worden ’multi-state’ HLPs gedefinieerd. Deze HLPs hebben geometrie attributen die bij verschillende toestanden horen, parameters delen en van elkaar afhankelijk kunnen zijn.
Analyse en afvlakking van een kabelboom   De mogelijkheden van bestaande programma's om kabelbomen te analyseren en af te vlakken (ook wel 'platslaan' genoemd) zijn beperkt. Het gebruik van deze systemen is tijdrovend, repetitief en bestaat voornamelijk uit handmatig werk. De automatische afvlakkingsmethode van de elektrische werkbank van het beschikbare CAD systeem is onbetrouwbaar en vereist extra verificaties en aanpassingen. In dit werk zijn procedures ontwikkeld om automatisch de 3D toestand van een kabelboom te analyseren ten opzichte van de 2D productietoestand voor overschrijdingen van buig- en twistlimieten, en om de aanwezigheid van niet af te vlakken uitbreekpunten te evalueren. De evaluatie van de flexibiliteit wordt uitgevoerd met behulp van regels gebaseerd op experimenten die gevalideerd zijn door jarenlange toepassing in de industriële praktijk bij Fokker Elmo. De resultaten worden aan de ingenieur gerapporteerd en zijn de belangrijkste invoerparameters voor het afvlakken. Een nieuwe afvlakkingsmethode voor kabelbomen is ontwikkeld die de 2D toestanden van kabelboom HLPs ten opzichte van elkaar oplijnt. De analyse- en afvlakkingsapplicatie genereert 2D kabelboom modellen met deformaties die binnen toegestane grenzen vallen en waarbij kabelbundels minimaal gedraaid zijn.

Fitten en configureren van het vormbord   Op dit moment wordt de lay-out van een platte kabelboom handmatig aangepast zodat deze binnen een standaard vormbordtafelmaat past. Dit is een zoekprobleem met meerdere kwaliteit- en productie efficiëntie doelstellingen zoals het minimaliseren van het tafelformaat, het elimineren van kruisingen en het optimaliseren van de productie ergonomie. Om er voor te zorgen dat de kabelboom vanuit de gefitte toestand vervormd mag worden naar de 3D installatie toestand, moeten ingenieurs op dit moment tijdrovende checks en aanpassingen uitvoeren. Om de noodzaak hiervoor te elimineren wordt het platte model gediscretiseerd in buigbare secties waarvan de buiglimieten worden berekend. Deze limieten beperken zowel een een ingenieur bij het handmatig fitten als automatische fit algoritmes gebruik makend van zoekmethodes. Dit elimineert de noodzaak voor tijdrovende controles en aanpassingen. Na met verschillende methoden voor automatisch fitten te hebben geëxperimenteerd, is de zogenaamde 'alignment' methode, gebaseerd op de heuristiek van handmatig fitten, geselecteerd en geïmplementeerd. Deze methode probeert bundels in takken te lijnen volgens een beperkte set van buig- en flipstrategieën. Een standaard tafelmaat wordt automatisch geselecteerd uit een lijst van standaard maten, de lay-out wordt aangepast ten behoeve van ergonomische eisen en productie instructies worden toegevoegd.
Samenvatting

Uitwisseling van CAD definities Het uitwisselen van CAD definities is een bekend probleem in multidisciplinaire, parallelle product ontwikkelomgevingen waar meerdere partijen bij betrokken zijn. Om de ontwikkelde KBE oplossingen te integreren in het ontwerpproces voor elektrische bekabeling wordt een aanpak voorgesteld en geïmplementeerd die gebruik maakt van een 'interpreter' KBE applicatie die de definitie van een 3D kabelboom in STEP AP214 formaat vertaald naar een HLP parameterisering. Deze aanpak biedt een adequate oplossing voor het integreren van externe, op maat gemaakte software ontwerpoplossingen in het conventionele ontwerp- en ontwikkelproces.

De potentie van KBE Om te bepalen of het gebruik van de KBE applicaties de doorlooptijd vermindert en voor vormborden zorgt van voldoende kwaliteit is een validatie uitgevoerd met 25 kabelbomen van een commercieel vliegtuigprogramma. De kwaliteit van de resulterende vormbord tekeningen is goedgekeurd door vormbord ontwerp experts. De benodigde tijd om een vormbord te maken wordt gereduceerd van uren tot minuten. Uit een conservatieve berekening blijkt dat in de praktijk een vijfvoudige toename in productiviteit te behalen is.

De ontwikkelde vormbord KBE applicaties bieden functionaliteiten die niet aanwezig zijn in conventionele CAD systemen. De KBE applicaties zijn gericht op specifieke technische processen die buiten de het toepassingsgebied van algemene CAD systemen (zullen blijven) liggen. De applicaties kunnen de hoeveelheid repetitief werk aanzienlijk verminderen, en tegelijkertijd verzekeren dat eisen met betrekking tot fysieke limieten en productierichtlijnen nageleefd worden.
Acknowledgements

These past four years of PhD research have gone by in the blink of an eye. I had the opportunity to work together with people from different organizations, and there are many who contributed in some way to this PhD work.

First of all I would like to thank my promotor and supervisor Michel van Tooren for his trust and support. I wish him the best of luck at his new position in South Carolina. I want to extend my gratitude to Gianfranco La Rocca for getting me interested in KBE, providing guidance and thorough feedback. I have very much enjoyed working with you ever since my job as a student assistant for the Aircraft Design course.

Thanks to my colleagues at the Faculty of Aerospace Engineering. In particular my roommates Gandert van Raemdonck, Durk Steenhuizen and Francesco Bertels for your companionship and interesting discussions. Zaoxu Zhu, it was great to have a fellow PhD researcher to talk to on the subject of aircraft wiring design. Thanks to all the other former and current staff members and PhD students of SEAD/FPP: Mark, Roelof, Arvind, Michiel H., Lin, Bettie, Nana, Haiqiang, Michiel S., Maurice, Reinier, Ali, Jan, Dipanjay, Cory, Marcel, Leo, Joris, prof. Torenbeek, Feijia, Xiaojia, Mo Li, Fengnian, Peijun, Hamid, Lex, Abhishek, Wim, and the rest. Even through all the changes our group has gone through in the last few years, it remained a very pleasant workplace.

Many thanks as well to all the students I had the privilege to supervise during their master thesis work: Joep van Heugten, Maarten Nelissen, Francesco Bertels, Jelle Veraa, Pieter-Jan de Witte and Pieter van Assche.

I would also like to thank AgentschapNL for funding the research and in particular Dik Verduin for his interest in the project.

This work would not have been possible without the support and funding by Fokker Elmo and active involvement of many of its senior experts. Thanks to: the 'Task
ACKNOWLEDGEMENTS

Force’ members Jaap Jobse, Karin Rens, Ad Rens, André Hagenaars, Bastiën van de Plasse and Bert Zwanenburg, your critical but positive attitude have been invaluable. I hope that the results of this work will in the near future help to alleviate your workload. The members of the 'Steering Committee’ Simon Taylor, Fons Bingen, Richard Moes, Joost Theunis, Joost van Leeuwen and Theo Tetteroo. Han Suykerbuyk and Folkert Stilma for our discussions on improving manufacturing methods an especially broadening my perspective of the electrical wiring market during our trip to the Productronica fair in Munich. Kees Nuyten and John van Lugtenburg for providing me an interesting working place at Fokker Elmo, your enthusiasm for KBE and being more than just colleagues. I am looking forward to our future photography trips.

The research project got off to a flying start, thanks to the support given by KE-works, in particular its founders Joost Schut, Jochem Berends and Stefan van der Elst. Your experience in practical knowledge engineering work at Fokker combined with your background as former PhD students was invaluable. Also to Maarten, Hermen, Dominik, Jacques, Peter, Stan, Tijl, Bastiaan and all the other KE-workers, thanks for including me in the great working environment KE-works has, including the lively Friday afternoon discussions, fussball matches and KE-onions.

I want to thank Dave Cooper from Genworks International for providing support with developing the KBE applications. I very much appreciate the hospitality you, Kai and the kids offered during my visit to Detroit.

A lot of thanks to Wouter Falkena and Bastiaan van den Berg for your interest in my work and for being such supportive friends and teammates. Justin Koning, no one is more fun to ‘talk planes’ with, I’m looking forward to our next roadtrip! I am deeply grateful to my family, Yme, Claire, Lotte, Pelle and Annick, for their unconditional support and confidence in me throughout all my studies. And most of all I want to thank Willemien for believing in me and for always being there for me.

Tobie van den Berg

Delft, October 21st, 2013
Curriculum Vitae

Tobie van den Berg was born on May 12th, 1983 in Nacala-Porto, Mozambique. He has lived a large part of his childhood in Africa, besides Mozambique also in Togo and Guinea-Bissau where he frequented French schools. In 1998 his family moved to Culemborg in the Netherlands where Tobie attended O.R.S. Lek en Linge and received his VWO diploma in 2001.

He started his studies in Aerospace Engineering at Delft University of Technology in 2001 to obtain his bachelor’s degree in 2006. During these years his studies were interrupted for one year to work as a board member of the student volleyball society on a full-time scholarship. Tobie started his master’s studies at the Design of Aircraft and Rotorcraft chair, where he also had different student assistant jobs, of which the most important was assisting in writing and lay-outing the book "Aerodynamics of Transport Aircraft". Tobie had an internship with ADSE in Hoofddorp. He obtained his master’s degree in 2009 for his work on a Knowledge Based Engineering (KBE) application for conceptual aircraft wing design.

As his graduation project sparked interest in engineering design automation, he accepted a PhD position to continue working on KBE applications. His work was performed as part of the F3 project, a Dutch national Strategic Research Program funded by AgentschapNL and Fokker Elmo. The aim of the project was to support the generation of electrical wiring harness manufacturing drawings at Fokker Elmo using KBE. The work was performed with support of and at Fokker Elmo in Hoogerheide, as well as KE-works in Delft. The research has resulted in three demonstrator applications able to largely automate the steps involved in generating manufacturing drawings, as presented in this dissertation. During his PhD research, he supervised master students not only from Aerospace Engineering, but also from the faculties of Industrial Design Engineering and Mechanical Engineering. He provided support for the Advanced Design Methodologies master’s course and participated in the faculty’s flight practical as a coordinator.

Tobie currently lives in Delft with his girlfriend Willemien.
An increase in productivity is required for the European aviation industry to remain competitive in a world with rising competition, shorter development time-scales, more complex products and decreasing numbers of technically skilled personnel. Fokker Elmo, a Dutch aircraft electrical wiring harness manufacturer, is facing these challenges. At present, preparing a wiring harness for manufacturing is a repetitive, time-consuming and mostly manual, experience-based process.

The research presented in this thesis aims at developing techniques to largely automate the generation of wiring harness manufacturing drawings, using Knowledge Based Engineering (KBE). A development approach is proposed as well as KBE building blocks. Three KBE applications are developed and tested that offer functionalities not present in general-purpose CAD systems. These applications can considerably reduce the amount of repetitive work, while ensuring compliance to physical constraints and manufacturing guidelines.