JOURNAL OF AIRCRAFT
Vol. 46, No. 6, November–December 2009

Knowledge-Based Engineering Approach to Support Aircraft Multidisciplinary Design and Optimization

Gianfranco La Rocca* and Michel J. L. van Tooren†
Delft University of Technology, 2629 HS Delft, The Netherlands

DOI: 10.2514/1.39028

This paper introduces the concept of the design and engineering engine, which is a modular computational design system to support distributed multidisciplinary design and optimization of aircraft. In particular, this paper discusses the architecture and the functionalities of the multimodel generator module, which is a knowledge-based engineering application developed to model the geometry of both conventional and novel aircraft configurations and to automate the generation of dedicated models for low- and high-fidelity analysis tools. This paper demonstrates the capability of the knowledge-based engineering approach to record and automate complex engineering design processes, such as the generation of models for finite element analysis. The time reduction gained by process automation, together with the enabled use of high-fidelity analysis tools earlier in the design process, constitute significant achievements toward a broader exploitation of the multidisciplinary design and optimization methodology, as well as the development of novel aircraft configurations.

I. Introduction

MULTIDISCIPLINARY design optimization (MDO) is recognized to be one of the most promising methodologies in the field of aircraft design and the design of complex products in general [1,2]. However, a number of technical and nontechnical barriers are constraining its transition from a promising to a consolidated and routinely exploited methodology. After almost 30 years of development in the field of applied optimization techniques, MDO still struggles to get the same attention of the traditional disciplines in the field of aerospace engineering. In the early years, the low-fidelity level of the analysis tools applicable in MDO frameworks and the low computational power were the main limitations [3]. In 1998, a review study carried out by the AIAA MDO technical committee [4] revealed that the exploitation of MDO was still hampered by the problematic use of high-fidelity tools. A multidisciplinary design and optimization approach was still only possible by using low-level fidelity analysis tools, although the use of computational fluid dynamics (CFD) and finite-element-method (FEM) tools was limited to tradeoff studies and monodisciplinary design cases. At that time, the use of genuine MDO methods within the industry at large was still rather limited and mostly confined to the detail design phase [5]. Including high-fidelity analysis tools in a MDO framework for conceptual design of complete aircraft configurations was not yet common practice [6,7]. The lack of robustness and flexibility of high-fidelity analysis tools still forced designers to use lengthy and mostly manual preprocessing activities, and the level of achieved automation was not yet sufficient to support the highly iterative nature of the MDO approach. However, the availability of MDO systems based on the use of accurate physics-based analysis tools was an absolute necessity to support the design of novel aircraft configurations, such as blended wing bodies [8] or PrandtlPlanes [9]. As elaborated by Bowcutt [10] and Morris [11], the high-level of integration in such designs required (and benefits the most from) a true multidisciplinary and optimization approach. The use of simple conceptual rules from the classical handbook design methods became useless due to the lack of any statistic base or experience from past aircraft development programs.

Advances in the exploitation of the MDO require the implementation of a different organizational structure in a company, as discussed in [12–14]. At the same time, the development of any new MDO framework must account for the possibility to support collaborative design. Discipline specialists from different groups, departments, and companies (often nongeographically collocated) should be able to collaborate within large design programs while still making use of their own well-trusted design and analysis tools.

On the basis of the previous considerations, it is clear that advances in the field of MDO do not depend exclusively on the availability of faster computers or more efficient optimization algorithms but on the overall development of computational frameworks geared toward flexibility, automation, and exploitation of high-fidelity analysis systems [15]. A nonexhaustive list of needs follows, which any computational design system aimed at supporting the MDO approach should be able to fulfill:

1) The system structure should be easy to adapt to different design cases (not limited to conventional aircraft configurations) and to the specific needs of the various phases of the design process.

2) The system should support the integration into one engineering design environment of different design, analysis, and optimization tools scattered across distributed and heterogeneous computer networks.

3) The system should be able to integrate both commercial off-the-shelf (COTS) and in-house developed design, analysis, and optimization tools.

4) The system should integrate both low- and high-fidelity analysis tools.

5) The system should guarantee the coherence and synchronization of the data/models used by the various disciplinary analysis tools involved in the design and optimization process.

6) The system should support automation of all the repetitive activities related to the iterative nature of the MDO approach, including those related to pre- and postprocessing of data and models and their transfer between the various design and analysis tools

II. Paradigm of the Design and Engineering Engine

Based on the requirements discussed in the previous section, the Design of Aircraft and Rotorcraft group of the Delft University of Technology started the development of a distributed computational design system concept called the design and engineering

Presented as Paper 967 at the 45th AIAA Aerospace Sciences Meeting and Exhibit, Reno, Nevada, 8–11 January 2007; received 10 June 2008; revision received 26 July 2009; accepted for publication 11 August 2009. Copyright © 2009 by Delft University of Technology. Published by the American Institute of Aeronautics and Astronautics, Inc., with permission. Copies of this paper may be made for personal or internal use, on condition that the copier pay the $10.00 per-copy fee to the Copyright Clearance Center, Inc., 222 Rosewood Drive, Danvers, MA 01923; include the code 0021-8669/09 and $10.00 in correspondence with the CCC.

*Assistant Professor, Aerospace Engineering, Design of Aircraft and Rotorcraft Division, Kluyverweg 1.

†Full Professor, Aerospace Engineering, Design of Aircraft and Rotorcraft Division, Kluyverweg 1. Member AIAA.

engine (DEE). The first conceptual development of the DEE was initiated within the framework of the European project MOB: a computational design engine incorporating multidisciplinary design and optimization for blended wing-body configuration [11]. Since then, it has kept advancing through a number of national research projects and not exclusive collaboration with the industry. As illustrated in Fig. 1, the DEE consists of a multidisciplinary collection of design and analysis tools, able to interface automatically and exchange data and information, to support what-if studies and MDO. The main purpose of the DEE is to support and accelerate the design, analysis, and optimization process of complex products like aircraft through the automation of noncreative and repetitive design activities [16–18].

The following main components constitute the DEE architecture:

1) The multimodel generator (MMG), which is a true knowledge-based engineering (KBE) application developed with the twofold intent of providing designers with a parametric modeling environment (to define generative models of conventional and novel aircraft configurations) and feeding various analysis tools with dedicated aircraft model abstractions (as required for the verification of the generated design). The MMG structure and functionality are discussed in more detail in this paper.

2) The initiator, which is actually a set of sizing tools, is able to provide an initial set of parameter values for the MMG. In fact, the MMG offers the possibility to instantiate an aircraft model based on a given set of parameter values, but it does not have any knowledge to select/calculate those values autonomously. Various parameter initiating tools have been (and still are being) developed (e.g., to size a fuselage given the payload requirements or to size the trapezoidal wing planform based on mission requirements). Within the scope of the initiator also falls the provision of a rough estimation of the mass and stiffness distribution in the aircraft structure, based on a preliminary estimation of the aerodynamic loads. Eventually, the scope of the initiator is to provide a feasible initial solution to initiate the multidisciplinary optimization process (see the feasilization process in [19]).

3) A suite of analysis tools (the discipline silos), which can be low- and/or high-fidelity analysis tools (for example, panel codes or CFD), either developed in-house or COTS. The set of analysis tools varies according to the design case at hand, yet it always operates on data and models generated by the MMG.

4) The converger and evaluator box checks the various analysis tools (e.g., the flow solver) to see if they have reached convergence, evaluates if the performance/characteristics of the design meets the objectives set by the designer, and defines the next parameter set when running an optimization problem.

5) The communication framework, represented in Fig. 1 by the set of connectors linking the various DEE components, takes care of the data and information flow between the various design and analysis tools and enables the overall design process sequence. The current framework, for which the agent-based architecture is described in [20], uses web technologies to have the DEE tools communicating with each other, even when located on different computer networks and running on machines with different operating systems.

To participate in the DEE structure, a software component must be able to operate autonomously while exposing an adequate input/output interface. In other words, it must be able to run in batch mode, providing remote accessibility and hands-off operation. These characteristics are essential to the implementation of the DEE modular architecture.

The design team should be facilitated in adapting and reconfiguring the design framework according to the nature of each design case, in incorporating new design and analysis capabilities, and in maintaining the system and keeping it up to date. The modular structure of the DEE offers these opportunities, in contrast to a tightly integrated design system. However, modularity comes along with the overhead of building proper/generic interfaces and, in general, some detrimental effect on the data exchange speed. Despite these negative aspects, modularity seems a proper investment that, so far, has resulted in expandability, upgradability, and exploitability of the distributed design environment [15].

Given the central role of the MMG, the DEE systems fits in the category of the so-called geometry-in-the-loop design systems. In this sense, the DEE differs from the traditional conceptual design systems in which geometry is generated just as final output of a computational process based on simple statistics and/or semi-empirical equations. The DEE approach differs also from the category of systems based on grid perturbation [21]. Different dedicated models are generated for each of the analysis tools involved in the DEE and not just one tessellated representation tailored to the main analysis code, as discussed in [22]. This yields the additional advantage that large model variations are allowed during design space exploration, whereas grid perturbation methods are generally limited to virtual sandpaper work.

The possibility of having multidisciplinary analysis tools operating on dedicated and consistent geometry models, systematically updated

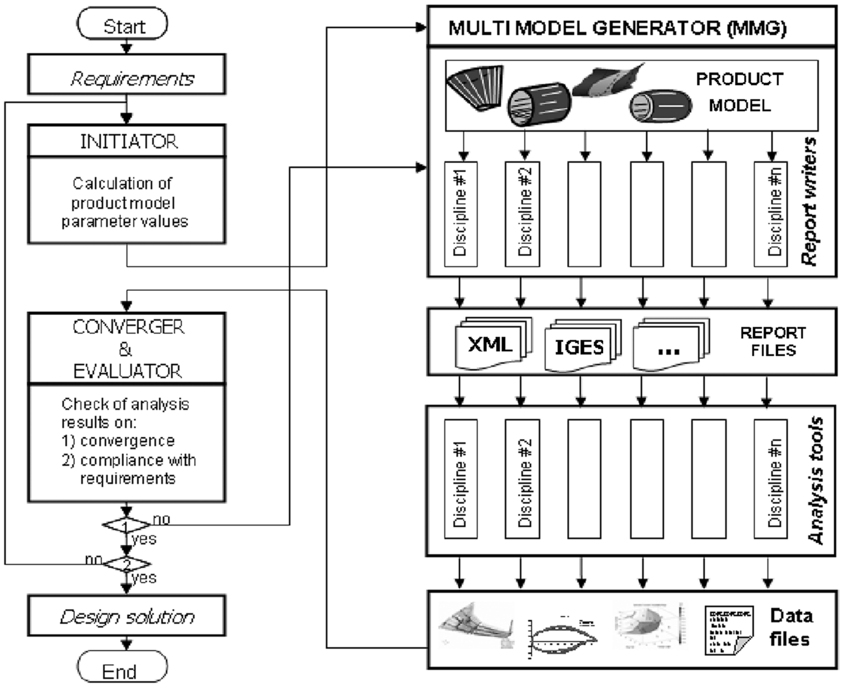


Fig. 1 Paradigm of the DEE.

at each parametric variation, makes the MMG the actual technological enabler of the whole DEE system. Indeed, advocates [10,22,23] of the geometry-in-the-loop approach indicate the geometry generation as the keystone to succeed and often the greatest impediment to integrated design.

In the following sections, role, structure, and functionality of the KBE MMG are discussed in detail. In particular, the capabilities of the MMG to instantiate models of aircraft (or isolated components such as wings and control surfaces) and generate specific sets of data and information to support automated finite element (FE) structural analysis will be addressed.

III. Structure and Functionality of the Multimodel Generator: High-Level Primitives and Capability Modules

The MMG, as shown in Fig. 1 (top right), has two main functional blocks, namely, the product model and the report writers. The product model represents the actual (geometry) modeling tool, which offers the designer the possibility to create instantiations of the aircraft concepts he/she has in mind. The MMG provides designers with a suite of parametrical blocks, so-called high-level primitives (HLPs), which can be easily shaped and assembled to build up a large number of aircraft configurations and variants. Examples of these primitives are the wing trunk, the fuselage trunk, the engine part, and the connection element (as shown in Fig. 2). Each primitive has been defined in the KBE environment as a class [24,25]. The designers, via an editable input file (the MMG input file), can assign different values to the given class attributes and call for multiple HLP instantiations, such that either conventional or novel aircraft configurations can be generated and then stretched/morphed into an infinite number of variants. A part of the parameter/attribute set used to define the wing-trunk HLP is shown in Fig. 2 (left).

The HLP approach offers a more effective way to follow the way a designer looks at an aircraft during the conceptual development phase than the approach offered by a conventional computer aided design (CAD) system. A designer sees the aircraft as an assembly of basic solutions to fulfill functionalities, such as generating lift and storing payload, not as an assembly of curves, surfaces, and points.

KBE, with its peculiar capability to merge CAD and object-oriented programming [26–28], has provided all the necessary ingredients to concretize the concept of HLPs into a working software application. The programming approach is what actually turns the HLPs into smart entities able to contain knowledge and reuse it systematically. For example, the HLPs know how to create/modify their shape based on a set of input parameters; they know how to connect to each other maintaining continuity and integrity (i.e., keeping outer surface and inner structure properly connected).

Furthermore, KBE offers the possibility to record, inside the HLPs, the knowledge to use the necessary workarounds that are needed to avoid and/or correct some of the typical geometry manipulation errors (e.g., missed or failed surface intersection operations) caused by the CAD engine embedded in the KBE system.

Even more relevant is the capability provided by the KBE approach to record the knowledge required to process the geometry features of the HLPs and generate dedicated input models (reports in the specific KBE system parlance) for the various analysis tools operating in the DEE, such as CFD and FEM tools. The report-writing capability actually represents a fundamental functionality of the MMG, and it is of paramount importance for supporting distributed multidisciplinary design.

Indeed, each one of the various discipline specialists involved in the design of an aircraft has a different and specific view on the aircraft under consideration. This view consists of a dedicated model including only those features that are relevant to his/her disciplinary domain and analysis tools, and all the rest is filtered out. For example, an aerodynamicist is not interested in the definition of the internal structure elements of the aircraft or in the positioning of the internal aircraft systems. On the other hand, a structure specialist does not care about specific aerodynamic features. Moreover, a structure specialist involved in the conceptual phase of the structure design is not interested, for example, in the shape of the stringers, flanges, or the positioning of the rivet holes, whereas a structure specialist involved in the detailed design of a wing panel is definitely interested in those details.

Eventually, models must be generated to satisfy the specific needs of all the specialists involved in the various design aspects. As a matter of fact, this is a critical bottleneck in the traditional design process. Many specialized models need to be generated, forcing designers and discipline specialists to go through lengthy and repetitive geometry preprocessing activities (typically frustrating work, prone to errors).

In general, all these models are generated by different actors, using different software tools, hence they are difficult to be maintained coherently. To tackle the problem of generating coherent models for multidisciplinary analysis, another set of classes has been developed called capability modules (CMs). Indeed, HLPs and CMs represent the main constituents of the MMG product model. Although the HLPs are mainly responsible for the generation of the geometric representation of the aircraft, the CMs have the capability to operate on such geometry and create dedicated abstractions suitable for the various DEE analysis tools.

Again, the KBE programming approach has been used to capture, in different CMs, the process knowledge [29] required to transform the geometry of the HLPs into (e.g., a cloud-of-points representation for aerodynamic analysis) sets of meshable surfaces for FE analysis (FEA) other model abstractions, as shown in the example of Fig. 3. Whatever is the number and shape of the HLP instantiations used to define a certain aircraft configuration, the CMs will be able to automatically process the geometry of each instantiation.

Figure 4 shows the example of a (half) blended wing-body aircraft (BWB) model, built up using five instantiations of the wing-trunk HLP. Each instantiation is transformed by the SurfaceSplitter CM into sets of meshable surfaces for FEA (i.e., properly cut fragments of skins, spar, and rib surfaces).

Once the CMs have processed the various HLP instantiations, all the generated data and information (both geometry- and non-geometry-related) are collected by the set of MMG report writers (Fig. 1) and exported to the various DEE disciplinary analysis tools by means of standard and/or dedicated file formats [30].

Whereas, the HLPs principally provide a service to the designer, the CMs mainly attend to the disciplinary specialists, allowing them

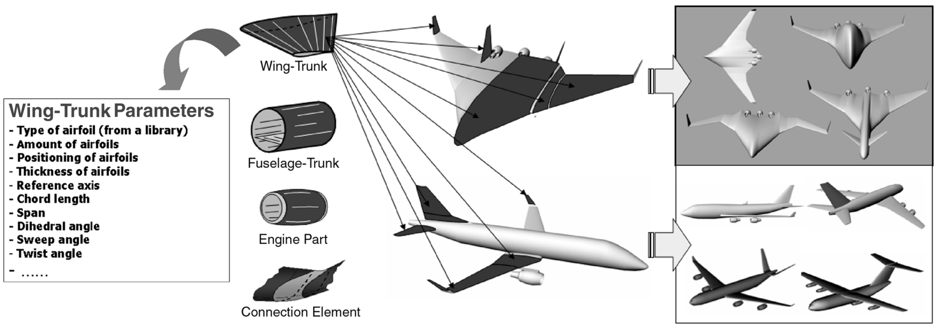


Fig. 2 The HLP buildup approach.

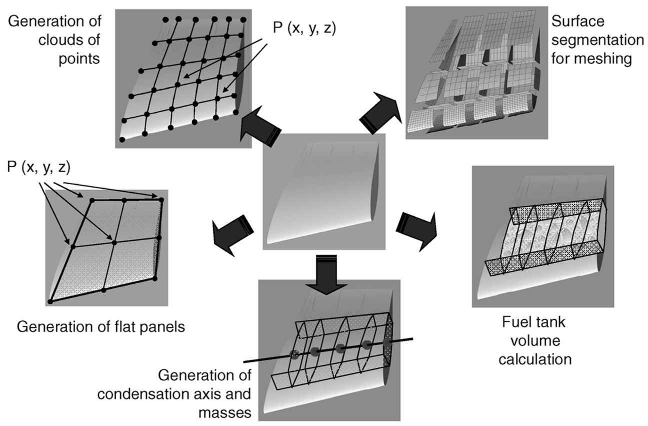

Generation of clouds of points
P (x, y, z)
Surface segmentation for meshing
P (x, y, z)
Generation of flat panels
Fuel tank volume calculation
Generation of condensation axis and masses

Fig. 3 CMs process a wing-trunk HLP instantiation into a set of model abstractions to support different analyses. From top left, counter clockwise, the following CMs are shown in action: PointsGenerator, PanelsGenerator, CondensedMassesGenerator, WingFuelTank, and SurfaceSplitter.

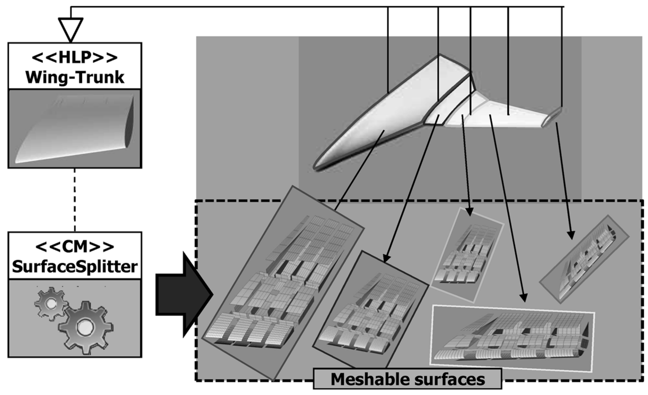


Fig. 4 The SurfaceSplitter CM processes all the wing-trunk HLP instantiations used to model a BWB into sets of meshable surfaces to support FEA.

to use their own analysis tools without the overhead of many lengthy and tedious preprocessing operations.

To offer the reader a deeper insight into the working mechanisms of HLPs and CMs, the next two sections specifically address the modeling process of a generic wing structure and the definition of the SurfaceSplitter CM. The link between the MMG and the PATRAN/NASTRAN FEA environment will be illustrated, showing how the KBE approach can enable the automation of the entire process: from the configuration of the aircraft structure to the launch of the structural solver.

IV. Parametric Definition of a Generic Wing Structure Configuration

The unified modeling language (UML) class diagram in Fig. 5 defines the MMG responsible for modeling and preprocessing any winglike system (e.g., a wing, a canard, a tail empennage, etc.). It shows the relations between the wing trunk, the connection element, and various CM classes. Four CMs are those previously illustrated in Fig. 3, namely, SurfaceSplitter, CondensedMassesGenerator, Wing-FuelTank, and PointsGenerator.

The wing-trunk and connection-element HLPs consist of two main components: one for the generation of the outer surface, and the other for the internal structure. Although a separated module, the inner structure of the HLP is defined associatively to the outer surface (i.e., when the external shape of the winglike system is modified, for example, by a change in airfoil selection, the shape of the various structural elements automatically adapts). Although not shown here, the same principle has been applied in the definition of the fuselage trunk and the relative structure.

Using the MMG structural design capabilities developed so far, designers can define any number of ribs, spars, stringers, frames, and floor elements and affect position and orientation of each one individually. Number, position, and orientation of each element are assigned using lists of parameters published in the MMG input file together with the parameters defining the aircraft outer surfaces. Figure 6 shows an example of a generic wing element generated by the MMG and the parameter definition of two ribs and one trailing edge (TE) riblet. In fact, the MMG allows different approaches for positioning and orienting the various structural elements, using as reference other structural elements, or reference vectors like the flight direction. In the example of Fig. 6, rib x is positioned on a plane that intersects spar 1 at 50% of the spar length and is oriented at 90 deg with respect to the direction of spar 2. Rib y is positioned on a plane that passes through the root of spar 2 (0% of the spar 2 length) and is oriented at 90 deg with respect to the same spar 2. Ribs and LE/TE riblets can be generated and oriented independently from one another. Partial ribs and runout spars (see the center spar in the example of Fig. 6) can be defined as well.

In addition, the user can define so-called virtual ribs and spars, which are support entities that can be used as a reference for other real elements. An important point is that, although a virtual spar or rib does not appear in the model, it actually affects the model for what concerns the segmentation of the surfaces (as discussed in the next section). Spars and ribs can also be defined as partially virtual elements. In the example of Fig. 6, the center spar starting at the root section and running out at the fourth rib actually becomes virtual after the fourth rib. Though the center spar of the example is not physically present in the area between the fourth rib and the wingtip section, it still causes the same skin and rib segmentation as a real full extending spar. Additionally, it can be used for positioning and orienting ribs as any normal spar.

V. Knowledge-Based Surface Segmentation to Support Finite Element Analysis Automation

To set up a FE model from the untrimmed geometry produced by the design department, the FE specialist will have to perform a lot of manual work to prepare the model geometry for meshing. The surfaces of all the model components (e.g., the skins, the spars, the ribs, and the riblets in the case of a wing model) have to be cut along their intersections in order to produce sets of meshable surfaces (i.e., surface segments which have no more than four edges, each edge matching with only one edge of the adjacent surface segments). Figure 7 shows two segmentations of a wing-box (WB): one is properly performed, and the other shows connectivity errors [31]. This so-called surface segmentation process is well known to be time expensive and often not trivial. Additionally, every time a change occurs in the model topology, the segmentation process has to be performed again. Unfortunately, only when all model surfaces are properly segmented will the automatic meshing functionalities provided by most of the FE preprocessors work.

Therefore, the segmentation process features all the characteristics of a good candidate for translation into a KBE application: length, repetition, and plenty of rule-based geometry manipulations. The FE specialist applies a set of mental rules and best practices when manually performing the segmentation process. By means of knowledge acquisition techniques [32,33], a large part of these rules and best practices has been elicited through interviews with FE experts and then captured inside the SurfaceSplitter CM. As previously illustrated in Fig. 5, given a generic aircraft model built with any number of HLPs instantiations, the SurfaceSplitter is able to process, one by one, all of the various HLP instantiations and finally deliver a set of surfaces that are suitable to be meshed (whatever the topology of the generic aircraft).

Figure 8 shows the use case elaborated during the development of the SurfaceSplitter module (note that it does not show the technical implementation of the segmentation process [34]) and includes a number of constraints/indications provided by FE specialists. Those indications (see the text in curly brackets in Fig. 8) form extremely valuable information because they reflect the mental scheme used by

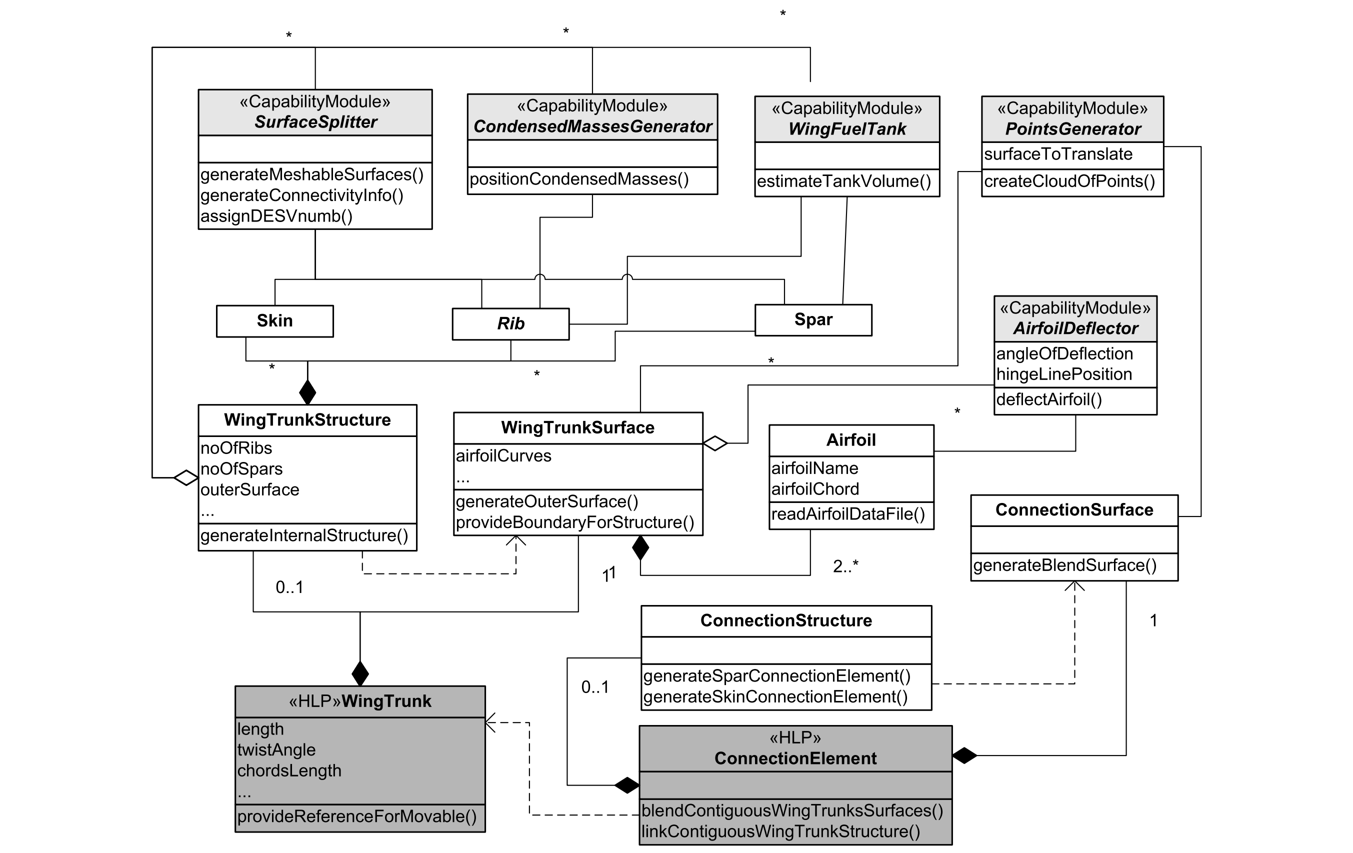

«CapabilityModule» SurfaceSplitter
generateMeshableSurfaces()
generateConnectivityInfo()
assignDESVnumb()
«CapabilityModule» CondensedMassesGenerator
positionCondensedMasses()
«CapabilityModule» WingFuelTank
estimateTankVolume()
«CapabilityModule» PointsGenerator
surfaceToTranslate
createCloudOfPoints()
Skin
Rib
Spar
«CapabilityModule» AirfoilDeflector
angleOfDeflection
hingeLinePosition
deflectAirfoil()
WingTrunkStructure
noOfRibs
noOfSpars
outerSurface
...
generateInternalStructure()
WingTrunkSurface
airfoilCurves
...
generateOuterSurface()
provideBoundaryForStructure()
Airfoil
airfoilName
airfoilChord
readAirfoilDataFile()
ConnectionSurface
generateBlendSurface()
ConnectionStructure
generateSparConnectionElement()
generateSkinConnectionElement()
«HLP»WingTrunk
length
twistAngle
chordsLength
...
provideReferenceForMovable()
«HLP» ConnectionElement
blendContiguousWingTrunksSurfaces()
linkContiguousWingTrunkStructure()
0..1
1
2..*
0..1
1
*

Fig. 5 UML class diagram of a generic winglike element, showing the modular structure of the wing-trunk and connection–element HLPs and their link to various CMs.

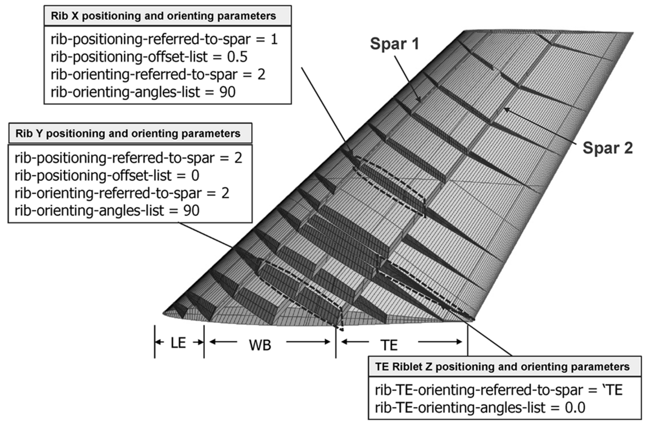

Rib X positioning and orienting parameters
rib-positioning-referred-to-spar = 1
rib-positioning-offset-list = 0.5
rib-orienting-referred-to-spar = 2
rib-orienting-angles-list = 90
Rib Y positioning and orienting parameters
rib-positioning-referred-to-spar = 2
rib-positioning-offset-list = 0
rib-orienting-referred-to-spar = 2
rib-orienting-angles-list = 90
Spar 1
Spar 2
LE
WB
TE
TE Riblet Z positioning and orienting parameters
rib-TE-orienting-referred-to-spar = 'TE
rib-TE-orienting-angles-list = 0.0

Fig. 6 Example of a MMG-generated wing structure configuration and input parameters used to define some ribs.

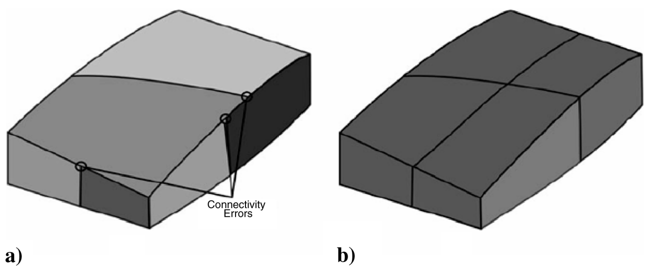


Fig. 7 Wing-box segmentation models a) with surface segmentation errors and b) properly segmented.

the FE specialists when performing the segmentation process by hand. However, indications, such as limit the number of extra generated surfaces or create triangles with the least possible sharp angles, are fuzzy. Knowledge acquisition techniques are used in order to transform those indications into explicit rules that are suitable for coverage by a software application. An example of a chunk of process knowledge that has been formalized and documented by means of a UML process diagram is presented in the next section.

Note that in Fig. 8, the use case for a system able to solve the segmentation problems automatically also includes the use case in which the designer is allowed to use virtual elements to fix the model segmentation by hand. The latter is a relevant use case because the designer must always have the opportunity to control and affect the segmentation process.

To show how the SurfaceSplitter CM works in practice, an example of the surface segmentation process applied to a generic winglike element is described here.

Figure 9 (top) shows a generic instantiation of a wing-trunk HLP. Whether such instantiation belongs to a BWB aircraft component (as those shown in Fig. 5), a conventional wing, a tail empennage, or a control surface is actually not relevant. In fact, the SurfaceSplitter CM operates on any HLP instance, independently from the instantiation purpose.

The wing trunk considered in this example has four spar elements (defining and confined to the WB area) and a number of ribs and riblets crossing the WB and/or the leading edge (LE) and (TE) areas. Note that some of these rib elements start and end at either a spar or the LE/TE line, whereas others start or end at the root or tip section of the given wing element (i.e., LE riblet 4, ribs 1, 3, 4, 5, and TE riblets 1 and 2). The latter are likely to cause troubles during the segmentation process.

Given the topology of this structure, the first operation performed by SurfaceSplitter is the intersection of the wing-trunk skins with all the spars and all the ribs. Also, each rib is intersected with all the spars and all other ribs and, finally, all the spars are intersected with all the

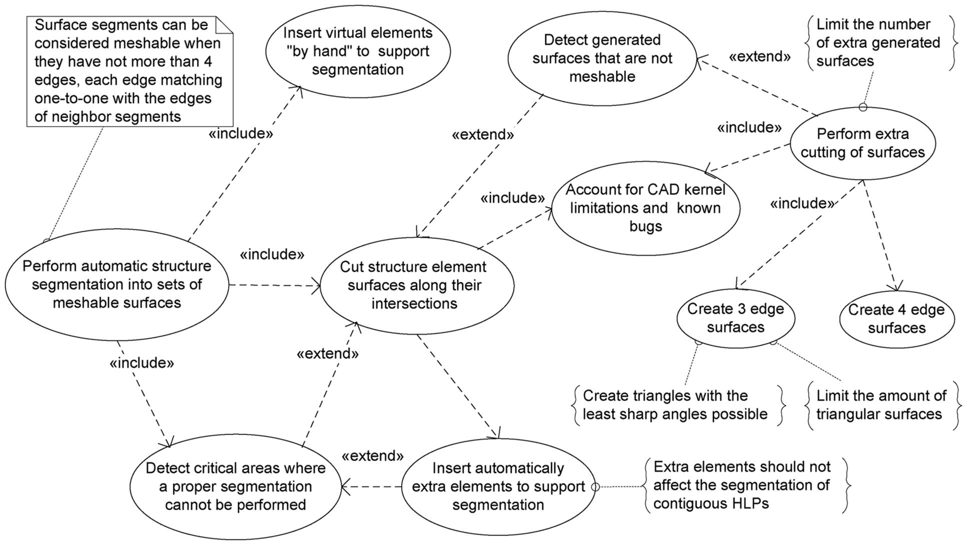

Surface segments can be considered meshable when they have not more than 4 edges, each edge matching one-to-one with the edges of neighbor segments
Insert virtual elements "by hand" to support segmentation
Detect generated surfaces that are not meshable
«extend»
Limit the number of extra generated surfaces
«include»
«extend»
«include»
Perform extra cutting of surfaces
«include»
Account for CAD kernel limitations and known bugs
«include»
Perform automatic structure segmentation into sets of meshable surfaces
«include»
Cut structure element surfaces along their intersections
Create 3 edge surfaces
Create 4 edge surfaces
«include»
«extend»
Create triangles with the least sharp angles possible
Limit the amount of triangular surfaces
«extend»
Detect critical areas where a proper segmentation cannot be performed
Insert automatically extra elements to support segmentation
Extra elements should not affect the segmentation of contiguous HLPs

Fig. 8 UML use case for the KBE system to perform the automatic surface segmentation of a generic geometric model.

ribs and all other spars (step 0 of the segmentation process). Possibly, some of these intersection operations do not produce any results, but this is handled by the CM without causing any runtime error. The result of the intersect operations is a set of spar, rib, and skin surface segments (which is subsequently scanned for nonmeshable surfaces). As highlighted in the example, Fig. 9 (middle), five nonmeshable skin elements (i.e., elements with more than four edges) are detected by the CM. As anticipated, they are caused by the ribs and riblets that either start or end at the root and tip sections of the wing trunk.

As a FE specialist would, SurfaceSplitter finds out that, within the skin panel delimited by spar 0 and 1, it is possible to fix at least one nonmeshable surface by forcing the generation of a virtual spar (step 1 of the segmentation process). As shown at the bottom of Fig. 9, the position of the extra virtual spar is automatically defined by the points where two ribs intersect the root and tip sections of the wing trunk. As introduced in Sec. IV, virtual spars do not appear in the wing model as actual structural elements, but they do affect the

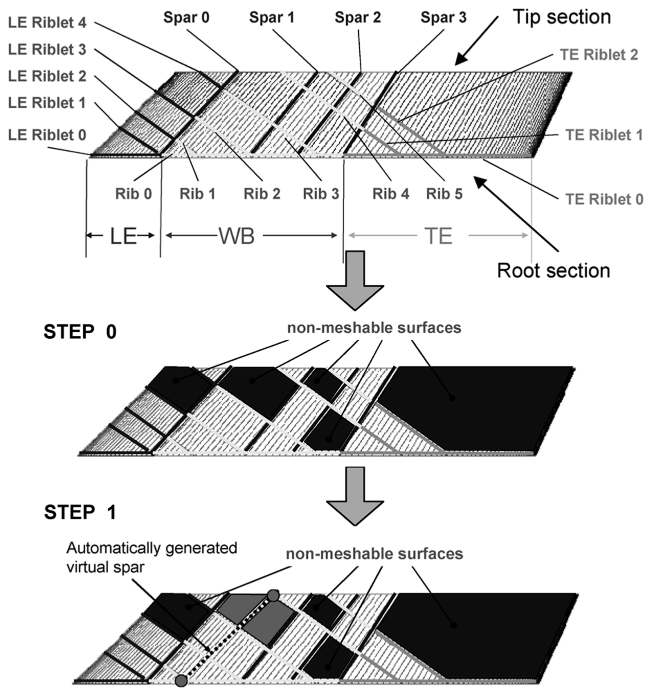


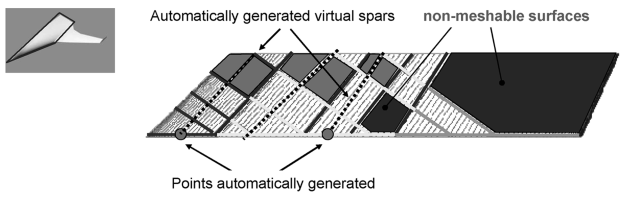
Fig. 9 Knowledge-based segmentation process of a generic winglike component (top). In step 0, skins, spar, and ribs are intersected with each other and nonmeshable surfaces are detected. In step 1, a first extra segmentation process resolves some nonmeshable surfaces by automating the generation of some virtual spars.

surface segmentation. The MMG functionality of automatic virtual spar generation to tackle segmentation issues can be switched off via the MMG input file. In this case, the designer is responsible for the manual definition (i.e., via the MMG input file) for the definition of the necessary extra cutting elements.

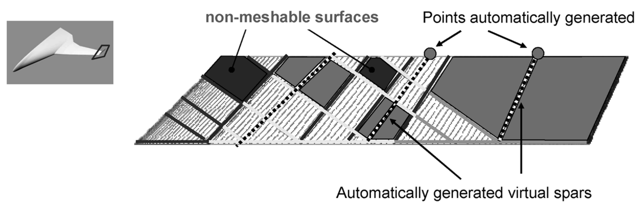
As shown at the bottom of Fig. 9, a single step of extra cutting element generation is not always sufficient to fix all the nonmeshable surfaces. In this specific case, there is only one couple of ribs (i.e., ribs 1 and 4) that can be used to define the position of the extra virtual spar. In fact, the generation of any other virtual spar to fix the remaining nonmeshable surfaces could possibly affect the segmentation of the wing trunks adjacent to the one under consideration (hence, it would increase the complexity of the overall segmentation process, as well as the total number of surface segments). Therefore, as a FE specialist would do, SurfaceSplitter checks the specific case and generates only extra virtual spars that do not affect the adjacent wing trunks (step 2 of the segmentation process). In Fig. 10a, SurfaceSplitter realizes that the wing trunk to be segmented (indicated with the left-side icons of Fig. 10) has an adjacent wing trunk at the tip section. Therefore, points are automatically generated on the free edge and used to generate extra virtual spars. As a result of this second segmentation step, two other nonmeshable surfaces get fixed. In Fig. 10b, SurfaceSplitter verifies that it is possible to generate support points for extra virtual spars only at the tip section of the given wing trunk. Also in this case, two other nonmeshable surfaces get fixed. In Fig. 10d, the presence of adjacent wing trunks, both at the root and tip sides, makes this second segmentation step useless. On the contrary, in the case of the isolated wing trunk (in Fig. 10d), this segmentation step is sufficient to obtain all meshable surfaces.

To tackle the segmentation of the remaining nonmeshable surfaces without affecting the adjacent wing trunks, a different segmentation approach is applied that is not based on the generation of extra virtual spars.

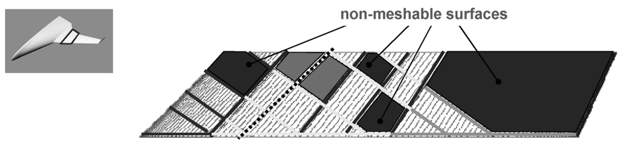
This extra segmentation method (step 3 of the segmentation process) consists of cutting, individually, each of the nonmeshable surfaces into two segments by using a cutting line passing through two noncontiguous vertices of the given nonmeshable surface. As illustrated in Fig. 11, all of the remaining nonmeshable surfaces from the previous two segmentation steps finally get fixed. As a FE expert would do, SurfaceSplitter uses this segmentation approach only as a last resource because it is likely to generate triangular surfaces that are not as good as quadrangles. In fact, there are more possibilities to split a pentagonal or hexagonal surface (surfaces with even more edges are not likely to occur) into two meshable surfaces. As an FE expert would do, SurfaceSplitter tries to generate the best combination of surface segments. Therefore, the combination of segments with the least sharp internal angles is selected in order to limit the generation of unhealthy FEs with a too-high aspect ratio.


Automatically generated virtual spars
non-meshable surfaces
Points automatically generated

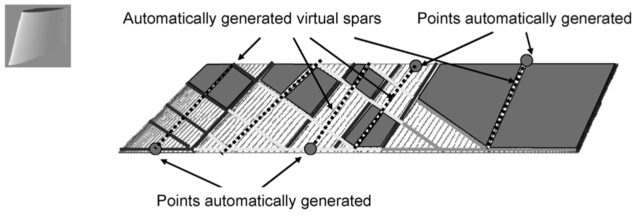
a)


non-meshable surfaces
Points automatically generated
Automatically generated virtual spars

b)

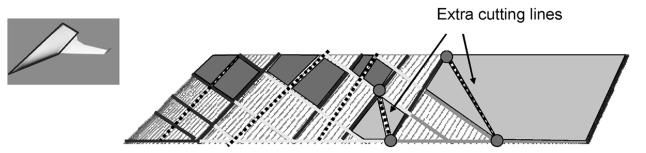


c)

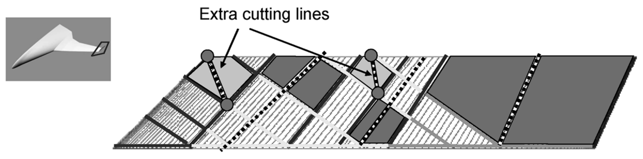

Automatically generated virtual spars
Points automatically generated
Points automatically generated

d)

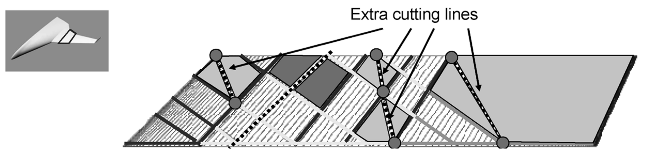
Fig. 10 Knowledge-based segmentation process of a generic winglike component. In step 2, a second extra segmentation process resolves some nonmeshable surfaces by generating extra virtual spars. The process is effective only when the extra segmentation is not perturbing eventual adjacent wing trunks (a and b). In the case of adjacent wing trunks, both at the root and tip section (c), the process is ineffective. In the case of isolated wing trunks (d), the process is completed successfully.



a)



b)



c)

Fig. 11 Knowledge-based segmentation process of a generic winglike component. Step 3: A third extra segmentation process resolves all the remaining nonmeshable surfaces by splitting them into two opportune segments, without affecting the segmentation of adjacent wing trunks.

The detailed process to select the best cutting approach has been formalized in the UML activity diagram of Fig. 12. This diagram gives also the idea of the level of knowledge formalization required to build (as well as to document) a KBE application.

Finally, in case a nonmeshable surface problem cannot be solved, due to (for example) an untrapped error of the CAD engine during geometry manipulations, the MMG will automatically label the given surface with a special nonmeshable tag and highlight it in red in the MMG graphical browser.

VI. Knowledge-Based Generation of Finite Element Models

The automatic surface segmentation process described previously is just one part of the total process required for the automatic generation of FE models. In this section, an approach for a seamless link between the MMG and the PATRAN/NASTRAN FE environment is described (for more technical details, refer to [31]).

Every time a surface segment is generated, the MMG attaches a special identification tag to it to record its membership (i.e., the HLP instance or the kind of structural element the given segment belongs to). To collect all the surface segments necessary to build up the FE model, a search routine scans the whole MMG product model and collects the various surface segments in groups according to their identification tags. Finally, these groups of surface segments are exported to the FE environment by using sets of initial graphic exchange specification (IGES) files (see Fig. 13).

Because the IGES format can transfer only geometric information, a complementary link is also created to export the relevant nongeometric information required to automate the FE model setup. For this purpose, the MMG automatically generates one lookup table for each surface segment that is exported via IGES (see the sketch in Fig. 13). The lookup tables, addressed here as FEM tables, report information such as thickness, material, membership identification, meshability, a list of nonstructural mass items to be attached, a design variable group, Cartesian coordinates of the corner nodes, and other attributes of all the surface segments generated by SurfaceSplitter.

The KBE–FE environment interface is com pleted with a smart PATRAN session file, based on the PATRAN command language (PCL), and a Python application to work around the low data handling capability of PCL [15].

The role of the session file is to automatically open a PATRAN database and import all the surface segments delivered via IGES files. Then, the Python application starts reading the content of the FEM tables. In particular, the Cartesian coordinates of each surface segment are compared with the Cartesian coordinates of the surfaces imported in the PATRAN database and, as soon as a match is found, all the information stored in the given FEM table is automatically mapped on the corresponding representation of the surface segment in PATRAN. As matter of fact, the coordinates of the corner points represent the biunivocal link between each surface segment generated by the MMG and its corresponding representation in PATRAN (see the sketch in Fig. 14).

After the information transfer from the MMG to the FEA environment is completed, the session file proceeds with the setup of the FE model. Material properties are assigned, nonstructural masses are positioned and attached to the given surface segments, boundary conditions are applied, the mesh is generated, and (finally) the analysis and/or optimization is performed using one of the supported solvers (e.g., NASTRAN or ABAQUS).

The Python application and the PATRAN/NASTRAN FE engine form together a kind of KBE application: Python provides the reasoning mechanism and the object-oriented features and PATRAN/NASTRAN provides the functional engine.

The implemented methodology to link the MMG with the FE modeling environment generates a powerful and seamless modeling

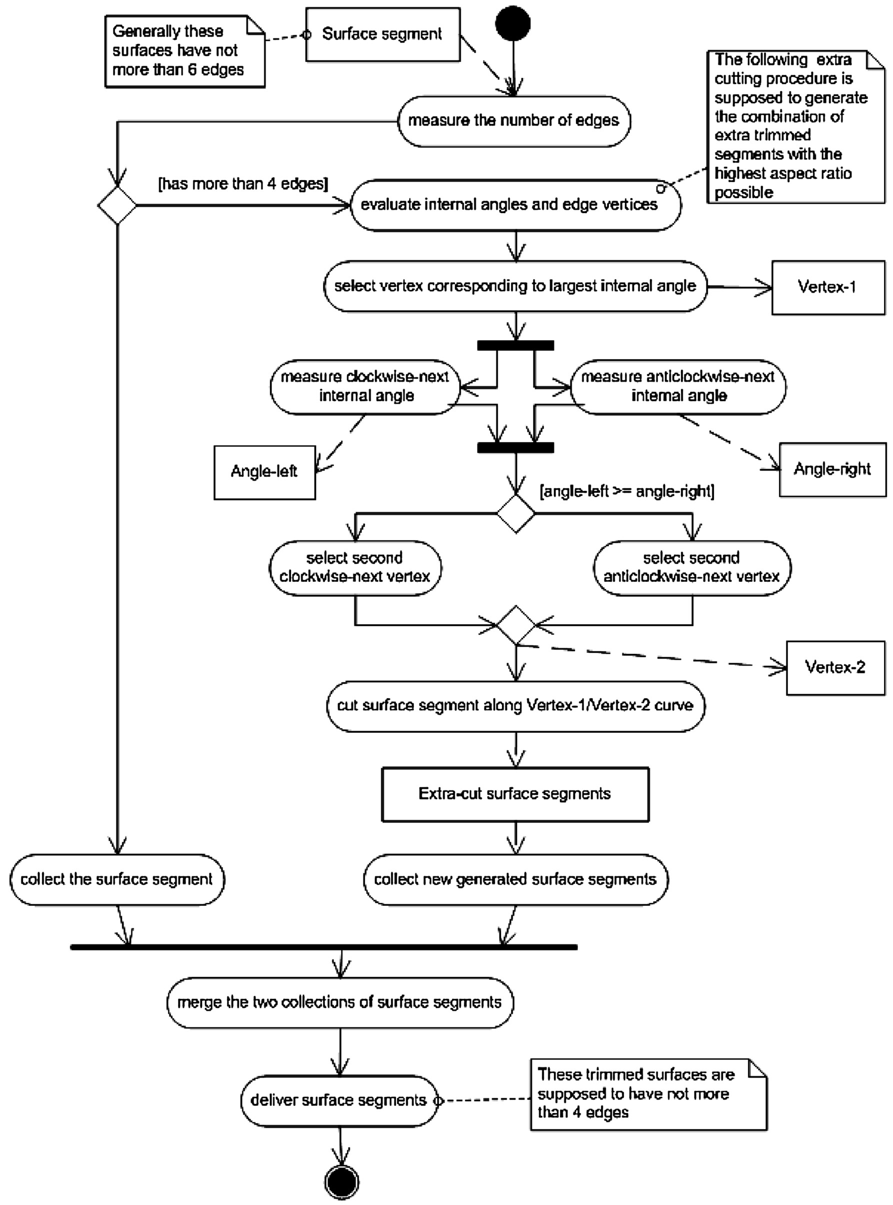
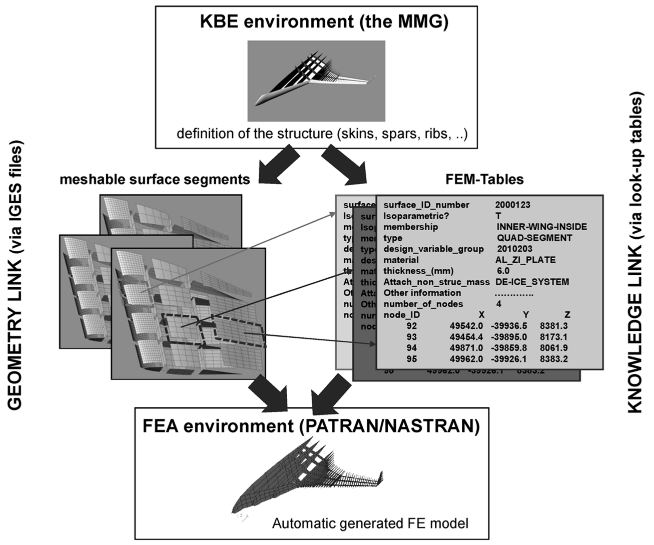

Generally these surfaces have not more than 6 edges
Surface segment
measure the number of edges
The following extra cutting procedure is supposed to generate the combination of extra trimmed segments with the highest aspect ratio possible
[has more than 4 edges]
evaluate internal angles and edge vertices
select vertex corresponding to largest internal angle
Vertex-1
measure clockwise-next internal angle
measure anticlockwise-next internal angle
Angle-left
Angle-right
[angle-left >= angle-right]
select second clockwise-next vertex
select second anticlockwise-next vertex
Vertex-2
cut surface segment along Vertex-1/Vertex-2 curve
Extra-cut surface segments
collect the surface segment
collect new generated surface segments
merge the two collections of surface segments
deliver surface segments
These trimmed surfaces are supposed to have not more than 4 edges

Fig. 12 Activity diagram detailing the extra cutting process to deal with surfaces segments that have more than four edges.


KBE environment (the MMG)
definition of the structure (skins, spars, ribs, ..)
meshable surface segments
FEM-Tables
GEOMETRY LINK (via IGES files)
KNOWLEDGE LINK (via look-up tables)
FEA environment (PATRAN/NASTRAN)
Automatic generated FE model

surface_ID_number	2000123		
Isoparametric?	T		
membership	INNER-WING-INSIDE		
type	QUAD-SEGMENT		
design_variable_group	2010203		
material	AL_ZI_PLATE		
thickness_(mm)	6.0		
Attach_non_struc_mass	DE-ICE_SYSTEM		
Other information	....		
number_of_nodes	4		
node_ID	X	Y	Z
92	49542.0	-39936.5	8381.3
93	49454.4	-39895.0	8173.1
94	49871.0	-39859.8	8061.9
95	49962.0	-39926.1	8383.2

Fig. 13 Seamless link between the MMG and the FEA environment. IGES files are used to transfer the geometry of the segmented surfaces; lookup tables (FEM tables) are used to transfer the information related to each surface segment and required to set up the FE model.

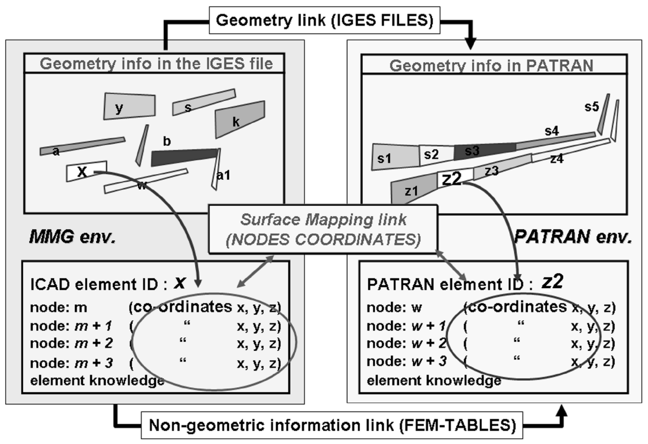


Fig. 14 The mapping process of the FEM table content is based on the match of the Cartesian coordinates of the corner points of the surfaces stored in the PATRAN database, with the Cartesian coordinates reported in the FEM tables.

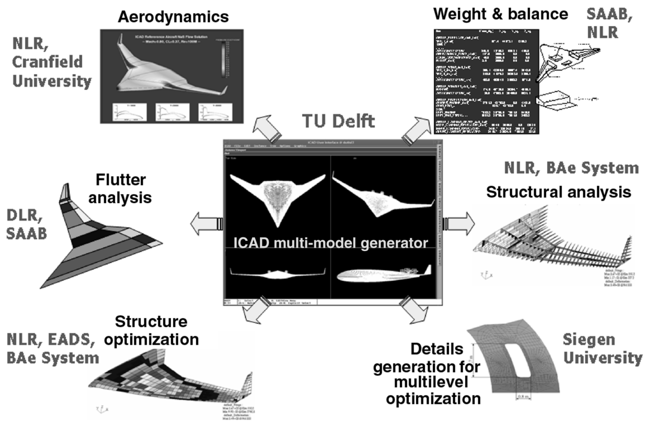


Fig. 15 **Role of the MMG within the MOB distributed MDO framework. The MMG provides dedicated models to a large set of distributed analysis tools, both low- and high-fidelity, and in-house developed and COTS.**

and analysis system, which allows the designer to evaluate many different design configurations, without worrying about a new FE model each time a variation is enforced in the shape or topology of the aircraft configuration.

VII. Results

A. MOB Blended Wing-Body Design Case

A successful validation case of the MMG concept is provided by the European project MOB on multidisciplinary design and optimization of BWB configurations [11,17,30]. Enabled by the MMG, the MOB consortium was able to address the design of an innovative complex aircraft configuration, for which no reference data or experience existed. Starting from a unique definition of a BWB aircraft configuration (Fig. 15), the MMG could extract a set of different (yet coherent) model abstractions, tailored to the analysis tools provided by the various partners from industry and academia. The commercial KBE system ICAD, by Knowledge Technology International, was used for the development of the MMG.

By means of the CMs, the MMG could generate not only models for FEA, but also models to support aerodynamic and aeroelastic analysis, and weight and balance assessment [35]. By means of specific code numbers automatically generated by the MMG and assigned via the FEM tables, all the structure segments could be grouped in opportune design variable areas before the launch of the structural optimization process [36]. Once the MOB computational framework was in place, more than 50 aircraft variants (including topological variations) were evaluated in just a couple of days (totally hands off), making use of high-fidelity analysis tools (including CFD and FE) running on a number of computers distributed across the multinational consortium. A traditional design approach, based on (partially) manual regeneration of the various models for each design loop, would have required months of work. Otherwise, it would have limited the study to a far lower amount of variants, or it would have prevented the use of high-fidelity analysis tools, with obvious consequences on the quality and reliability of the obtained results.

B. Vertical Tail Redesign Study

A DEE to support the redesign of the vertical tail of a passenger aircraft has been developed in collaboration with the Loads and Aeroelastics group at Airbus Germany. The scope of the project was the development of a computational system to facilitate a fast assessment of different tail design options, such as material and planform shapes [37]. The challenge was the rapid generation of aeroelastic and structural models of adequate fidelity, which are usually not available in the parametric way required to perform what-if studies in the preliminary design phase; neither could be built in a sufficiently short time. The implemented design process is sketched in Fig. 16 and can be summarized as follows. First, the MMG is used to model the configuration of the original vertical tail and to

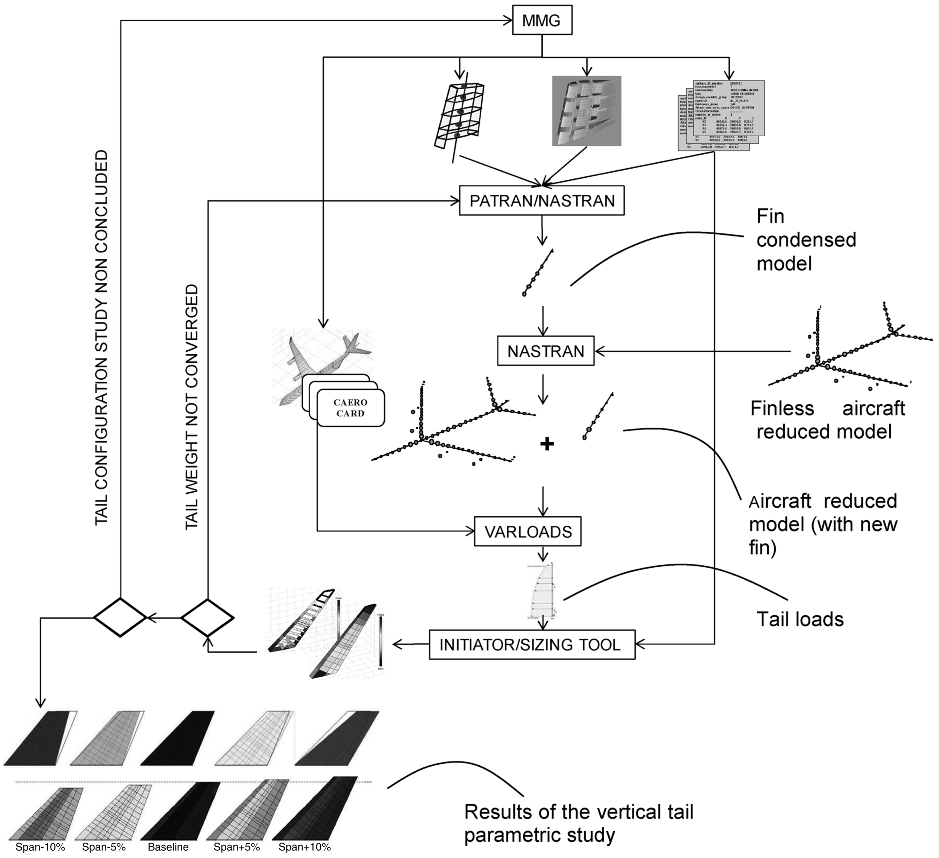

MMG
PATRAN/NASTRAN
Fin condensed model
NASTRAN
Finless aircraft reduced model
CAERO CARD
Aircraft reduced model (with new fin)
VARLOADS
Tail loads
INITIATOR/SIZING TOOL
TAIL CONFIGURATION STUDY NON CONCLUDED
TAIL WEIGHT NOT CONVERGED
Results of the vertical tail parametric study
Span-10%
Span-5%
Baseline
Span+5%
Span+10%

Fig. 16 **Schematic process of a vertical tail parametric study: the MMG feeds the tools for condensation, analysis, and sizing with different but coherent models.**

automatically generate the relative sets of segmented surfaces and FEM tables, which are then exported to the PATRAN/NASTRAN environment. Here, a PCL module is used to have NASTRAN generate a reduced model of the tail by condensing the structural and nonstructural fin masses on a set of dedicated (condensation) points, generated by the MMG CM CondensedMassesGenerator (see Fig. 3). The reduced model of the new tail is then connected to the previously generated reduced model of the original aircraft (less the vertical tail). The obtained reduced model of the complete aircraft model is then fed to VarLoads [38], a load analysis tool developed by Airbus, in which a dynamic yawing maneuver simulation is performed to generate the resulting tail loads. To operate the vortex lattice method implemented in VarLoads for the estimation of aerodynamic loads, the MMG is also responsible for the generation of a flat panel representation of the tail (which is provided to VarLoads) directly in the format of Nastran CAERO cards. The estimated tail loads are subsequently taken by the DEE initiator/ sizing tool [19] and mapped from the condensed mass model to another simplified tail model (automatically generated using the coordinates of the tail surface corners stored in the FEM tables), which is used to size the tail structural components and produce a weight estimation. The condensation and sizing process is then iterated until weight convergence, which is generally achieved within 5–7 loops of 45 min each. The overall process is then repeated for the next tail configuration to be studied (e.g., with a different span or sweep angle value). Hence, a new consistent set of segmented surfaces, FEM tables, condensation points, and CAERO cards is generated by the MMG, and the DEE framework takes care of the automatic coordinated execution of the entire instantiation and analysis process without any action required by the user.

VIII. Conclusions

Although MDO is recognized to be one of the most promising design methodologies in the field of aircraft design, several issues still prevent its full exploitation. The development of adequate generative modeling systems that are able to hook up to heterogeneous and distributed sets of analysis tools (high- and low-fidelity, in-house developed and legacy codes) and the automation of the manual work that typically hampers the analysis processes both deserve major attention and investment.

The DEE concept under development at Delft University addresses both of these issues. The DEE is a modular, loosely integrated design system that can accelerate aircraft multidisciplinary analysis and optimization by automating the typical lengthy and repetitive activities involved in the engineering design process. The MMG module represents the technological enabler of the DEE concept and makes use of KBE technology to automate many routine tasks and support the systematic application of the best design practices.

The inherent capabilities of KBE to integrate object-oriented rule-based design with parametric CAD were exploited to develop the main functional components of the MMG: the HLPs and the CMs. The HLPs, with their modular and parametric structure, provide designers with the level of flexibility and robustness needed to model a large number of aircraft configurations and configuration variants, including novel concepts. The CMs capture the process knowledge required to automate the typical preprocessing operations required to set up models for various analysis tools, both in-house developed and COTS, and of both low- and high-fidelity.

In particular it has been demonstrated that one of the greatest design challenges to date can be met by means of KBE (i.e., the automatic generation and modification of FE models) both for complete aircraft and components. By the systematic application of engineering rules elicited from specialists and codified into software modules, the DEE is able to set up and adjust FE models of complete aircraft (and components) independently of the vehicle configuration and its internal structural layout.

The enabled use of complex high-fidelity analysis tools in the early stages of the design process is not only important to increase the level of confidence in the characteristics of the designed product. It is a fundamental step toward the development of innovative aircraft configurations, in which semi-empirical statistics-based methods fall short, and a MDO approach based on a first principle analysis is key.

Acknowledgments

Financial support for this research has been partially provided by the European Commission under the Growth Programme for the research project, "MOB: A Computational Design Engine Incorporating Multi-Disciplinary Design and Optimisation for Blended Wing Body Configuration" (Contract Number G4RD-CT1999-0172), and by the Dutch Technology Foundation STW for the research project, "Parametric Modelling and Meshless Discretisation Methods for Knowledge-Based Engineering Applications" (Contract Number DLR.6054).

References

[1] White Paper on Current State of the Art, AIAA Technical Committee on Multi-Disciplinary Design Optimization (MDO) , AIAA, 1991.

[2] Bartholomew, P., "The Role of MDO within Aerospace Design and Progress Towards an MDO Capability Through European Collaboration," AIAA, 7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, Paper AIAA-98-4705, 2–4 Sept. 1998.

[3] Kroo, I., "Multidisciplinary Optimization Application in Preliminary Design: Status and Directions," 38th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference and Exhibit, AIAA Paper 1997-1408, April 1997.

[4] Giesing, J. P., and Barthelemy, J. M., "A Summary of Industry MDO Applications and Needs," 7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, AIAA, Paper 1998-4737, Sept 1998.

[5] de Weck, O., Agte, J., Sobieszczanski-Sobiesk, J., Arendsen, P., Morris, A., and Spieck, M., "State-of-the-Art and Future Trends in Multidisciplinary Design Optimization," 48th AIAA/ASME/ASCE/ AHS/ASC Structures, Structural Dynamics, and Materials Conference, AIAA, Paper 2007-1905, 2007.

[6] Alonso, J. J., "Requirements for MA&O in the NASA Fundamental Aeronautics Program," 12th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, AIAA, Reston, VA, https://info.aiaa.org/tac/ETMG/MDOTC/Keynote%20Presentations/Alonso%20MAO%202008.pdf, 2008.

[7] Bhatia, K. G., "A Perspective on Optimization," 12th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, AIAA, Reston, VA, https://info.aiaa.org/tac/ETMG/MDOTC/Keynote%20Presentations/Bhatia%20MAO%202008.pdf, 2008.

[8] Liebeck, R. H., "Design of the Blended Wing Body Subsonic Transport," *Journal of Aircraft*, Vol. 41, No. 1, 2004, pp. 10–25. doi:10.2514/1.9084

[9] Frediani, A., "The PrandtlPlane," *Proceedings of the International Conference on Computational and Experimental Engineering Sciences (ICCES 04)* [CD-ROM], Tech Science Press, Duluth, GA, July 2004, pp. 19–31.

[10] Bowcutt, K., "A Perspective on the Future of Aerospace Vehicle Design," 12th AIAA International Space Planes and Hypersonic Systems and Technologies, AIAA, Paper 2003-6957, Dec. 2003.

[11] Morris, A. J., "MOB: A European Distributed Multi-Disciplinary Design and Optimisation Project," 9thAIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimisation, AIAA, Paper 2002-5444, Sept. 2002.

[12] Raj, P., "Aircraft Design in the 21st Century: Implications for Design Methods," 29th AIAA Fluid Dynamics Conference, AIAA, Paper 1998-2895, June 1998.

[13] Belie, R., "Nontechnical Barriers to Multidisciplinary Optimisation in the Aerospace Industry," 9th AIAA/ISSMO Symposium of Multidisciplinary Analysis and Optimisation, AIAA, Paper 2002-5439, Sept. 2002.

[14] Malone, B., "On the Financial Impact of MDO on the Corporation," 9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimisation, AIAA, Paper 2002-5495, Sept. 2002.

[15] La Rocca, G., and van Tooren, M. J. L., "Enabling Distributed Multi-Disciplinary Design of Complex Products: A Knowledge Based Engineering Approach," *Journal of Design Research*, Vol. 5, No. 3, 2007, pp. 333–352.

[16] Staubach, J. B., "Multidisciplinary Design Optimisation, MDO: the Next Frontier of CAD/CAE in the Design of Aircraft Propulsion

Systems," AIAA/ICAS International Air and Space Symposium and Exposition, AIAA Paper 2003-2803, 14–17 July 2003.

[17] La Rocca, G., and van Tooren, M. J. L., "A Modular Reconfigurable Software Modelling Tool to Support Distributed Multidisciplinary Design and Optimisation of Complex Products," *16th CIRP International Design Seminar* [CD-ROM], College International pour la Recherche en Productique, Kananaskis, AB, Canada, 2006.

[18] van Tooren, M. J. L., Nawijn, M., Berends, J. P. T. J., and Schut, E. J., "Aircraft Design Support Using Knowledge Engineering and Optimisation Techniques," 46th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, AIAA Paper 2005-2205, April 2005.

[19] Schut, E. J., and van Tooren, M. J. L., "Design 'Feasilization' Using Knowledge-Based Engineering and Optimization Techniques," *Journal of Aircraft*, Vol. 44, No. 6, 2007, pp. 1776–1786. doi:10.2514/1.24688

[20] Berends, J. P. T. J., and van Tooren, M. J. L., "An Agent System Co-operating as a Design Build Team in a Multidisciplinary Design Environment," 44th AIAA Aerospace Science Meeting and Exhibit, AIAA Paper 2006-1482, Jan. 2006.

[21] Samareh, J. A., "Aerodynamic Shape Optimization Based on Free-Form Deformation," 10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, AIAA Paper 2004-4630, Aug. 2004.

[22] Vandenbrande, J. H., Grandine, T. A., and Hogan, T., "The Search of the Perfect Body: Shape Control for Multidisciplinary Design Optimization," 44th AIAA Aerospace Science Meeting and Exhibit, AIAA Paper 2006-928, Jan. 2006.

[23] Carty, A., and Davies, C., "Fusion of Aircraft Synthesis and Computer Aided Design," 10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, AIAA Paper 2004-4433, Aug. 2004.

[24] Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., and Lorensen, W., *Object-Oriented Modeling and Design*, Prentice–Hall, Englewood Cliff, NJ, 1991.

[25] Sully, P., *Modelling the World with Objects*, Prentice–Hall, Englewood Cliff, NJ, 1993.

[26] Cooper, D. J., and La Rocca, G., "Knowledge-Based Techniques for Developing Engineering Applications in the 21st Century," 7th AIAA Aviation Technology, Integration and Operations Conference, AIAA Paper 2007-7711, Sept. 2007.

[27] Cooper, S., Fan, I., and Li, G., "*Achieving Competitive Advantage Through Knowledge Based Engineering: A Best Practice Guide*," The Department of Trade and Industry, Cranfield, UK, 2001.

[28] Milton, N., *Knowledge Technologies*, Polimetrica, Monza, Italy, 2008.

[29] Rhem, A. J.,*UML for Developing Knowledge Management Systems*, Auerbach, Boca Raton, FL, 2006.

[30] Morris, A. J., Arendsen, P., La Rocca, G., Laban, M., Voss, R., and Hönlinger, H., "MOB: A European Project on Multidisciplinary Design Optimisation," *Proceedings of the 24th ICAS* [CD-ROM], International Council of the Aeronautical Sciences, Stockholm, Sept. 2004.

[31] Nawijn, M., and van Tooren, M. J. L., "Automated Finite Element Analysis in a Knowledge Based Engineering Environment," 44th AIAA Aerospace Science Meeting and Exhibit, AIAA Paper 2006-947, Jan. 2006.

[32] Shreiber, G., Akkremans, H., Anjewierden, A., de Hoog, R., Shadbolt, N., Van de Velde, W., and Wielinga, B., *Knowledge Engineering and Management: The CommonKADS Methodology*, MIT Press, Cambridge, MA, 2000.

[33] Milton, N., *Knowledge Acquisition in Practice: A Step-by-Step Guide*, Springer–Verlag, London, 2007.

[34] Booch, G., Rumbaugh, J., and Jacobson, I., *Unified Modeling Language User Guide*, Addison Wesley Longman, Reading, MA, 2005.

[35] La Rocca, G., Krakers, L., and van Tooren, M. J. L., "Development of an ICAD Generative Model for Blended Wing Body Aircraft Design," 9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimisation, AIAA Paper 2002-5447, Sept. 2002.

[36] Laban, M., Arendsen, P., Rouwhorst, W. F. J. A., and Vankan, W. J., "A Computational Design Engine for Multidisciplinary Optimisation with Application to a Blended Wing Body Configuration," 9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimisation, AIAA Paper 2002-5446, Sept. 2002.

[37] Cerulli, C., Schut, E. J., Berends, J. P. T. J., and van Tooren, M. J. L., "Tail Optimization and Redesign in a Multi Agent Task Environment," 47th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, AIAA Paper 2006-2241, 2006.

[38] Hofstee, J., Kier, T., Cerulli, C., and Looye, G., "A Variable, Fully Flexible Dynamic Response Tool for Special Investigation (Var-Loads)," *Proceedings of the CEAS/AIAA/NVvL International Forum on Aeroelasticity and Structural Dynamics*, Nederlandse Verening voor Luchtvaarttechniek, Amsterdam, The Netherlands, 2003.