DOI: 10.13009/EUCASS2023-465
Aerospace Europe Conference 2023 – 10TH EUCASS – 9TH CEAS

A Multidisciplinary Modeling System for Designing Fuselage Structures: Extending the Multidisciplinary Modeler

Bas van Manen†, Tobie van den Berg*, Ton van der Laan*, Bram Timmer***

* *GKN Aerospace – Fokker Aerostructures B.V.*
Anthony Fokkerweg 9, 3351 NL Papendrecht, Netherlands
** *GKN Aerospace – Fokker Elmo B.V.*
Aviolandalaan 33, 4631 RP Hoogerheide, Netherlands

Bas.vanManen@fokker.com – Tobie.vandenBerg@fokker.com –
Ton.vanderLaan@fokker.com – Bram.Timmer@fokker.com
† Corresponding Author

Abstract
The Multidisciplinary Modelers (MDM) package is a Python package to design and analyze wing-like structures. In this paper, the extensions made to the MDM package to enable the design and analysis of fuselage structures are described. It is first investigated how the structural layout components of wing-like and fuselage-like structure are different. Subsequently, the generic applicability of the current MDM package is investigated and the additions made to the MDM package are described. In the end, a general overview of the fuselage addition is given and the applicability of the new model is described in two design cases.

1. Introduction

As a Tier 1 specialist in lightweight structures, GKN Fokker Aerostructures receives requests for proposals and requests for trade studies from Original Equipment Manufacturers (OEM) [1]. With the current trend in Urban Air Mobility (UAM) vehicle development, it is crucial that these studies are performed as fast as possible, while also investigating multiple configurations instead of one. For this, GKN Fokker Aerostructures developed the Multidisciplinary Modelers (MDM) package [2]. The aim of the MDM package is firstly to reduce the design lead-time by increasing Design Automation (DA) and Multidisciplinary Design Analysis (MDA). Secondly, the MDM package should enable the use of Multidisciplinary Design Optimization (MDO) and Design Space Exploration (DSE) techniques in the design process. Lastly, the MDM package enables the transition to a front-loaded design process where most of the MDO study is performed before the project is executed.

The MDM package is developed to design wing-like structures, such as wings, flaps and movables. While the package proves to be already successful in industry, a need for modelling other products is noticed. For example, a fuselage structure, a tip-to-tip wing structure or even a fully integrated tail structure. The MDM package is developed in a generic matter, meaning that the package can be extended for these other products with only minimal changes to the existing code base. This generic construction of the package is exploited to develop the aforementioned needs for trade studies.

The aim of this paper is to describe one of the efforts made to extend the MDM package, enabling the design and analysis of fuselage-like structures. Extending the MDM package for fuselage structures is seen as the first step to enable the design and analysis of fully integrated structures (such as a fuselage with integrated wing or a fully integrated empennage structure). This is done by firstly describing the currently available MDM package. Next, the differences between fuselage-like and wing-like structures are investigated. The reusability of the MDM package and the additions made to the MDM package are described afterwards. The architectural layout of the fuselage extension is given and the practical use of this extension is described in two design cases.

DOI: 10.13009/EUCASS2023-465

2. The Multidisciplinary Modelers package

The MDM package is a Knowledge Based Engineering (KBE) application [2]. By definition, KBE is a methodology used to capture and implement knowledge-based rules into a functional application [3]. To enable this, Model Based Systems Engineering methods are used. Because both the product definition and the analysis methods are integrated into one KBE model, there is no need for case-specific workflow to be set up. This is often a time-consuming process, which is now fully integrated and automated.

The MDM package is built in an object-oriented way using the central product model approach. This means that there is one central product model and all information exchange between different product and analysis models happens through this central product model. The general structure of a workflow, file-based product model and a KBE product model is given in figure 1.

The MDM package is built with the Python programming language, using the KBE Software Development Kit (SDK) *ParaPy* [4]. The package consists of structural building blocks, developed into classes and functions. These building blocks are named primitives. A primitive provides a geometrical definition of a component, as well as links to different disciplinary models. A product model can be constructed from these primitives.

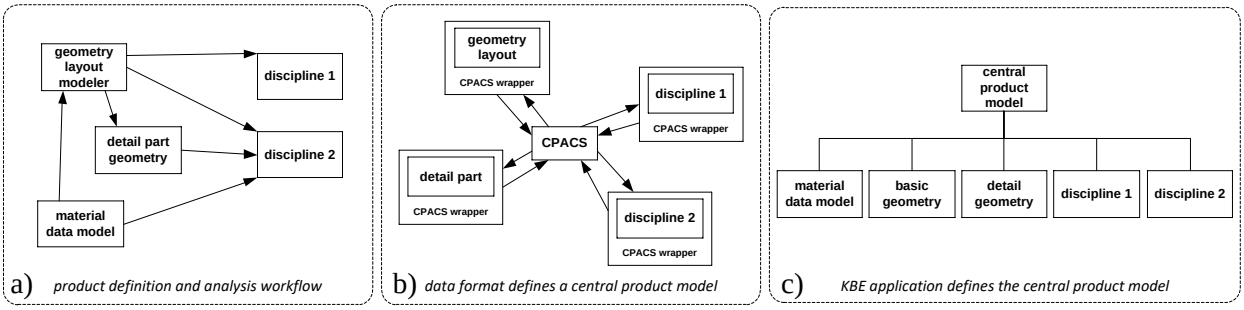

Figure 1: a) workflow, b) file-based product model, c) KBE product model [2]

As presented in [5], three products are currently available. These are the wing-box, movable and flap. The product model consists of one of these products, the required product definition inputs and the disciplinary models. This total product model is called a generator class and currently three of these generator classes are available: *WingboxGenerator*, *MoveableGenerator* and *FlapGenerator*. These generator classes enable the design and analysis of wing boxes, movables and flaps, respectively. Figure 2 schematically shows the different classes in a product model for a moveable. The distinction is made between product definition input, product definition, loads definition and product disciplinary models. At time of writing, the product manufacturing definition and product requirements are not yet implemented.

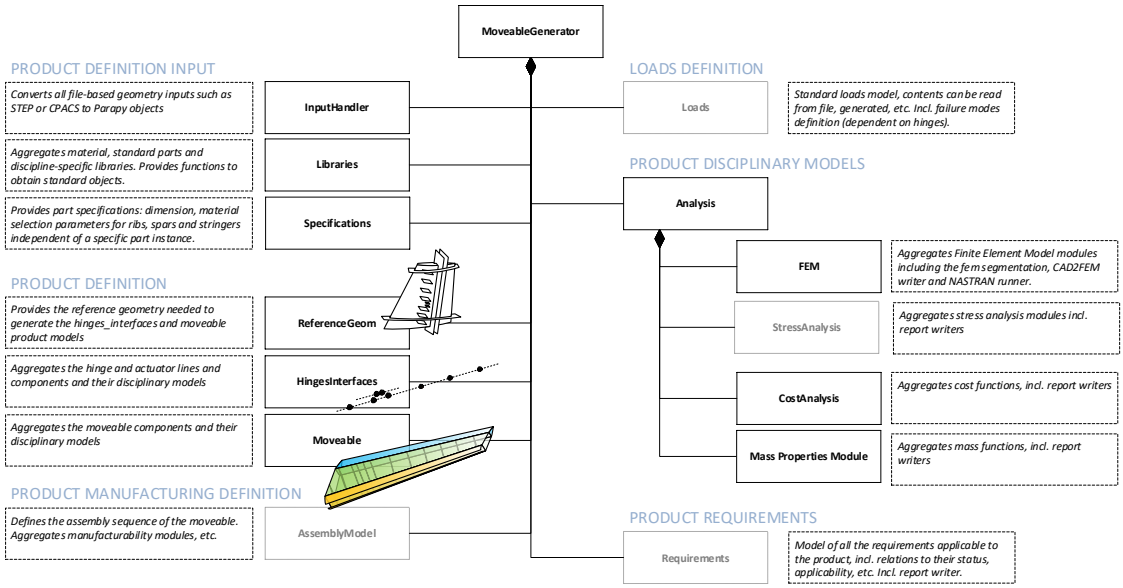

Figure 2: High-level architecture of the *MoveableGenerator* [2]

DOI: 10.13009/EUCASS2023-465

All primitives available in the package consist of disciplinary models. In this model, the interface is made between the definition model and the disciplinary tool. Because of this interface, providing the input to the disciplinary tool can be automated and analyses for an individual primitive can be performed automatically. Individual primitive disciplinary models are collected on generator level, where analyses on the entire product model can be performed. Currently, the following disciplinary models are present for each primitive [5]:

- Mass properties model: in-house Python-based tool for mass properties estimations
- Project-based cost model: in-house (proprietary) Excel tool to estimate part and assembly cost
- Open-source cost model: open-source Python-based cost estimation tool CATMAC [6]
- FEM segmentation model: fully segmented surface model of the geometry, to be used for FEM analysis
- CAD2FEM model: in-house PATRAN pre- and postprocessor tool for CAD geometries
- FEM model: ParaPy and Salome based mesh provider to be used in FEM analysis
- Stress model: in-house stress tool to calculate reserve factors of the structures given a certain load

At time of writing, developments have been made to integrate a manufacturing model in the central product model in order to take manufacturing aspects into account in the design process [5]. Furthermore, new generator classes are in development to enable design and analysis of tip-to-tip wings (including center interface) and fuselage structures. The MDM package is also tried to be applied in a multi-architecture optimization environment, where the automated disciplinary models can be exploited to fully automate a multi-architecture trade study [7].

3. Extending the Multidisciplinary Modelers package

While the current MDM package provides a generic approach to design and analyze structures, the available primitives are constructed with wing-like structures used as reference. Furthermore, structural components not present in a wing-like structure are not developed in the package yet for the same reason.

Because of this, extending the MDM code package can be divided into two categories: adapting the existing codebase and adding new code to it. This section will describe the efforts made to extend the MDM package by first explaining the conceptual differences (and similarities) between wing-like and fuselage-like structures. Afterwards, concrete examples will be given in how these items are tackled in the package.

The concept of wing and fuselage structures

In order to fully understand what is required to extend the MDM package to enable the design and analysis of fuselage-like structures, the conceptual differences between wing and fuselage structures is investigated. This research has two objectives. On one hand, it is investigated how a wing and a fuselage consist of similar components. Shared components mean that the existing codebase can be used to configure those structural parts. On the other hand, structural differences between wing and fuselage need to be investigated. New primitives are required for structural parts that are not present in a wing, but are present in a fuselage. Lastly, a primitive can already be present in a wing structure, but can be used in a fuselage when slight modifications are made.

An example of a typical aircraft structural layout is given in figure 3. Please note that this is a general example to visualize the typical components of a wing and fuselage structure, but this does not always have to be the case. In general, a wing consists of skin panels, spars, ribs and stringers. A fuselage generally consists of skin panels, frames and stringers. On top of that, a fuselage can have horizontal floor beams and struts to support the floor. For each structural component, different types can be present. For example, a frame can be closed (like a bulkhead), a frame can have a hole or a frame can be completely open (like a circumferential stiffener). Stringers can also have different shapes, such as L-stringers, hat stringers and blade stringers. These types of stringers can be present in both a wing and in a fuselage.

DOI: 10.13009/EUCASS2023-465

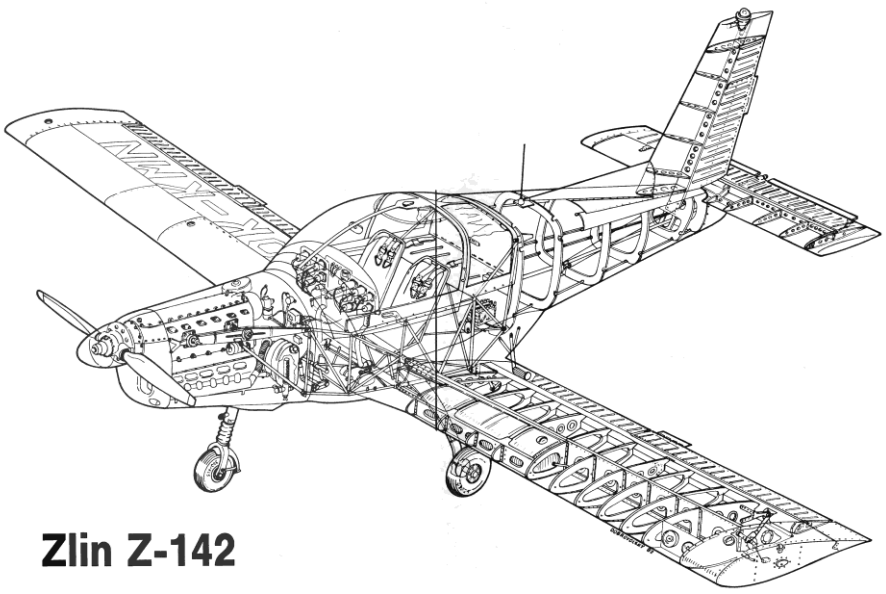

Figure 3: Structural layout of a general aviation aircraft. (ZLIN aircraft, 1990 [8])

Based on this information, several conclusions can be drawn on how the codebase should be extended. Firstly, different types of stringers are present in both a wing and in a fuselage. This means that the current stringer primitive should be adapted such that it can be used in both wings and fuselages. Secondly, both structures have skin panels, but they differ more in shape and orientation. It is to be investigated to what extend these primitives can be generically applied. Lastly, frames are structural components that cannot be found in wings. Therefore, a frame primitive must be added in which the structural behavior of a frame can be modelled.

Generic applicability of existing codebase

As mentioned earlier, the intention of the package is to be generically applicable to multiple products. A distinction can be made to the structural product definition and product definition inputs. This concept is visualized in figure 2. Where the structural product definition differs between wing-like structures and fuselage-like structures, the product definition inputs are close to identical. This means that definition inputs such as the libraries and input handler can directly be used with no additions or modifications. Moreover, the existing specifications library can also be used for primitives that are shared between wings and fuselages. This means that no modifications are necessary in the existing specifications library. The specifications will, however, be extended with the newly developed primitives that can be found in fuselage structures. These primitive specifications will not be made available for wing structures, but specifically only for fuselage structures. The product definition inputs are defined in a *BaseGenerator* class, which is specialized into a *BaseFuselageGenerator* and a *BaseWingGenerator*. In these specialized generator classes, structure-specific definitions can be stored. The class diagram of the base generator classes is show in figure 4.

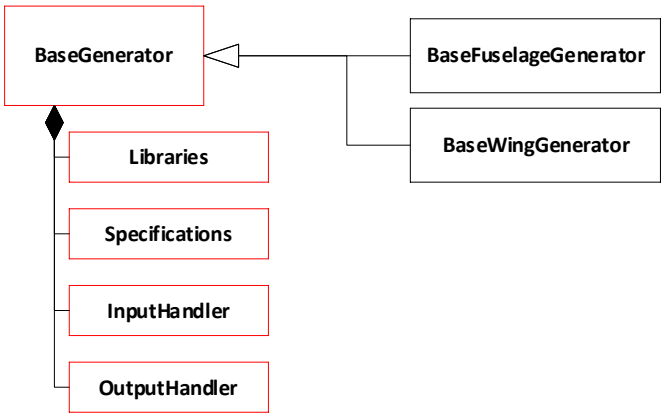

Figure 4: Class diagram of the base generator definition. Classes outlined in red already existed.

DOI: 10.13009/EUCASS2023-465

The existing codebase can be reused to define a fuselage structure in terms of product definition as well. In the previous section, it is concluded that the stringers used in wings and fuselages are identical. It is therefore tried to reuse the primitive for (wing) stringers completely in the fuselage structural product. Although the geometrical definition and disciplinary models of the stringers are similar, the position and orientation configuration is found to be different. By generalizing the existing stringer classes, most of the similar information can be reused. The generalization of the stringer class is visualized in figure 5. Information about the section specification and material choices is found in the generalized stiffener classes. The stiffener classes also consist of disciplinary models dependent on the material and shape of the section, such as the cost and mass model (not visualized in the figure). The stringer and fuselage stringer classes consist of information about the position and orientation of the stringer with respect to the OML, as well as orientation-specific disciplinary models, such as the FEM model and the stress model.

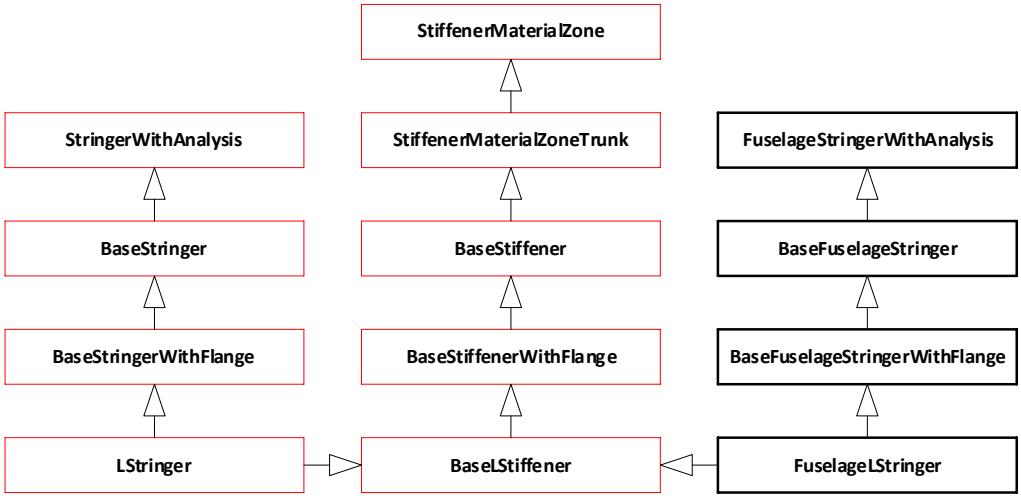

Figure 5: Class inheritance of the stringer primitive. Classes outlined in red already existed.

Two approaches are taken in order to reuse the interfaces to disciplinary models as much as possible. These approaches are visualized in figure 6, where the implementation of new primitives and the implementation of new types of material zones is shown. In this figure, (…) represents the name of a yet-to-be-implemented primitive. To show how the disciplinary models are currently set up, the spar primitive is also visualized in this figure. As visible in this figure, the implementation of the disciplinary models for a material zone is not changing. Hence, when a new material zone trunk is made for a new primitive, the corresponding disciplinary models are automatically working. Examples of disciplinary models that are on material zone level are the mass properties model and the open-source cost model. Disciplinary models that are based on the entire primitive cannot be implemented that way. For example, a *BasicSpar* is a *BasicSparWithAnalysis* where the primitive-dependent disciplinary models are implemented, as visible in the figure. These are, for example, the C2F model and the FEM model. When implementing the analysis class for a new primitive, the superclass for a *PartWithAnalysis* can be used. This class implements the base features and dependency tracking capabilities of a disciplinary model. Therefore, only the primitive-specific analysis classes need to be developed and the implementation in the disciplinary model tree structure happens automatically.

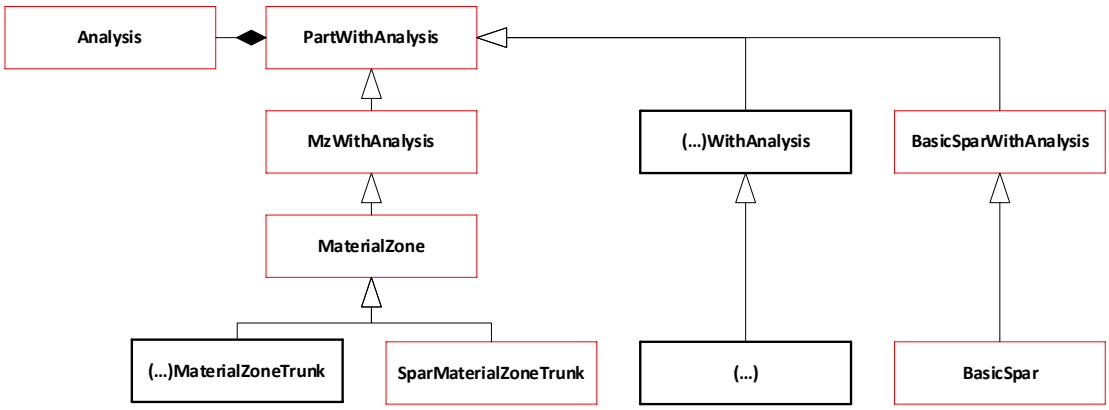

Figure 6: Class inheritance of a part with analysis. Classes outlined in red already existed.

DOI: 10.13009/EUCASS2023-465

Extending the primitives library

Structural parts that can be found in fuselages, but not in wing structures can be seen as new primitives. These primitives are added to the package as part of the fuselage extension. Although the structural part of the primitive is completely new, the generic base setup of the primitive class should be equal to the existing primitives. This also means that the base classes used for existing primitives also form the base for the new primitives. An example of a new primitive that is added as part of the fuselage extension is the frame. Frames can be defined as structural elements along the circumference of the fuselage.

In figure 7, the primitive classes required in the frame example are visualized. This figure consists two different type of frames, which are a closed frame (or bulkhead) and a basic frame. Both frame primitive consists of a web and a flange. The different type of frames have the same flange, but different type of webs. The primitive classes are added to the existing MDM package, as well as the required super classes. Both frame primitives share the same super classes. Frame-specific material zone information for example is stored in the *FrameMaterialZone* class, which is shared between the different types of frames. All newly added classes are subclasses of already existing base classes. Therefore, the core attributes and inputs for each primitive are kept equal between all primitives (existing and new).

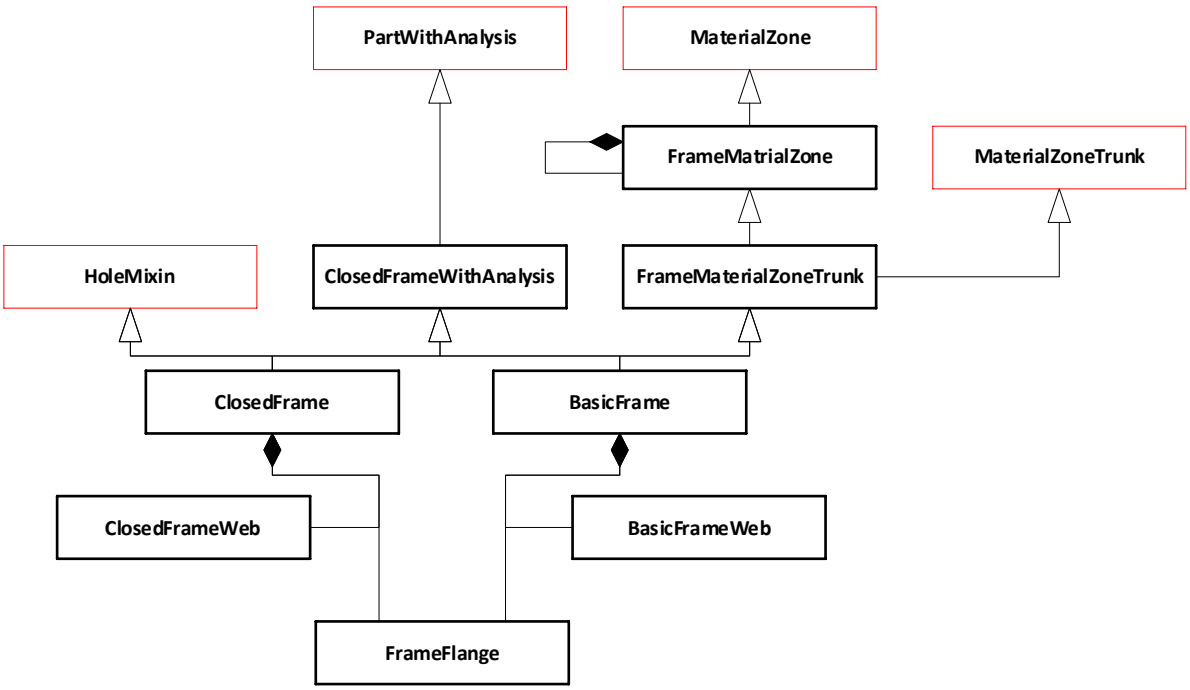

Figure 7: Class inheritance of a part with analysis. Classes outlined in red already existed.

4. The *FuselageGenerator* Architecture

In the extended MDM package, new product classes are also defined. On component level, the *Fuselage* class is added. On generator level, the *FuselageGenerator* class is added. This section will describe the architectural layout of the new generator and product classes. Based on UML class diagrams, new and existing classes will be highlighted. Furthermore, the different analyses present for the fuselage will be shown.

Architectural layout

Although the *FuselageGenerator* is a new generator product, the architectural layout of the generator class shall be similar to the currently implemented classes for wings, flaps and movables. This means that the top-level architecture will consist of a product definition input, a product definition and product disciplinary models. Product definition inputs are specified in a *Base* class and kept equal between the different generator classes. The *BaseGenerator* class specifies the material libraries, primitive specifications, input (file) handler and output (file) handler.

The product definition class of the fuselage is added in the extension of the MDM package. The *Fuselage* class consists of the structural layout of the fuselage and structural primitives of the fuselage. The structural primitives are

DOI: 10.13009/EUCASS2023-465

part of a station, which can be a lengthwise or a circumferential station. These lengthwise and circumferential stations can be subdivided into primary and secondary stations. Therefore, the fuselage product consists of the following parts:

- *Primary lengthwise stations:* These stations are perpendicular to the center line of the fuselage. The position of a primary station is explicitly defined. For example, the aft pressure bulkhead or the frames at the edge of the center wing box.
- *Secondary lengthwise stations:* The orientation of secondary lengthwise stations is similar to primary lengthwise stations. However, the position of secondary lengthwise stations is not explicitly defined, but with respect to the position of primary lengthwise stations. For example, frames with equal spacing in the center fuselage section between the forward and aft pressure bulkhead.
- *Primary circumferential stations:* These stations are in lengthwise direction, defined as normal to a lengthwise station section. A primary circumferential station position is defined as an explicit position along the section circumference. For example, the stringer below a window or the stringer at the edge of a skin panel.
- *Secondary circumferential stations:* The secondary stations are again positioned with respect to primary circumferential stations. The stations are also placed along the section circumference. For example, equally spaced stringers along the fuselage cross-section.

Where the stations are positioned is defined in the *FuselageStationsLayout*. As mentioned before, the primary station positions are explicitly defined and therefore the amount of primary stations is known. The number of secondary stations is not known beforehand and is determined in the stations layout class. At the moment, primary lengthwise stations are defined at fractions of the center line and secondary lengthwise stations are defined with a pitch between primary stations or with an absolute number between primary stations. Primary circumferential stations are defined at either a fraction of the circumference, at an absolute distance from the circumference reference point or at an angle from the circumference reference point.

The general layout of the newly added *FuselageGenerator* class is visualized in figure 8. Classes outlined in black are newly added classes as part of the fuselage extension. Classes outlined in red were already part of the MDM.

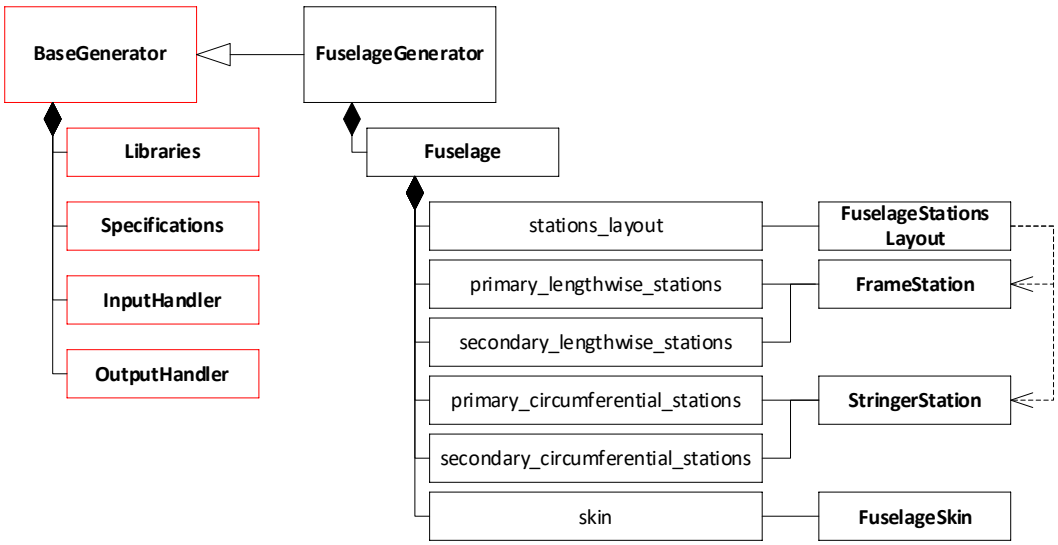

Figure 8: UML diagram of the fuselage generator class. Classes outlined in red were already existing.

Disciplinary models

In [5], it is described how disciplinary models are linked to individual primitives and collected on generator level. For existing primitives, the disciplinary models did not change. Disciplinary models for the new primitives are developed using the same methodology as the existing primitives. The collector on generator level of the fuselage, as well as the product interfaces on generator level are new developments.

DOI: 10.13009/EUCASS2023-465

At time of writing, the implementation of disciplinary models is not started yet. From the existing MDM package, *Base* classes can be used to set up the new primitive analysis classes and corresponding disciplinary models.

5. Design Cases and Scenarios

The extensions described in this paper are performed in the DEFAINE project [9] and the Dutch Mobility Fund project [10]. Design cases and scenarios where the fuselage modeler can be used are therefore obtained from these projects. From the DEFAINE project, a low-fidelity design case is described. In this case, only the reference geometry and structural layout are important. The Mobility Fund design case describes the design of a tail-fuselage section of a small air vehicle. This is a high-fidelity example, where part design and analysis are both considered.

DEFAINE: Low-fidelity design case

In one of the design cases used in the DEFAINE project the design of an Unmanned Aerial Vehicle (UAV) is considered. The main objective of this design case is to demonstrate the technologies developed within the DEFAINE framework. One particular tool used during the design case, the Electrical Wiring Interconnection System (EWIS) Architecture Modeler (AM), will generate a large number of EWIS channel architectures across which routing between systems will be performed. The channels from this channel architecture are the high-ways across which power and signals between systems can be routed, taking into account redundancy and separation criteria. An example channel architecture is shown in Figure 9.

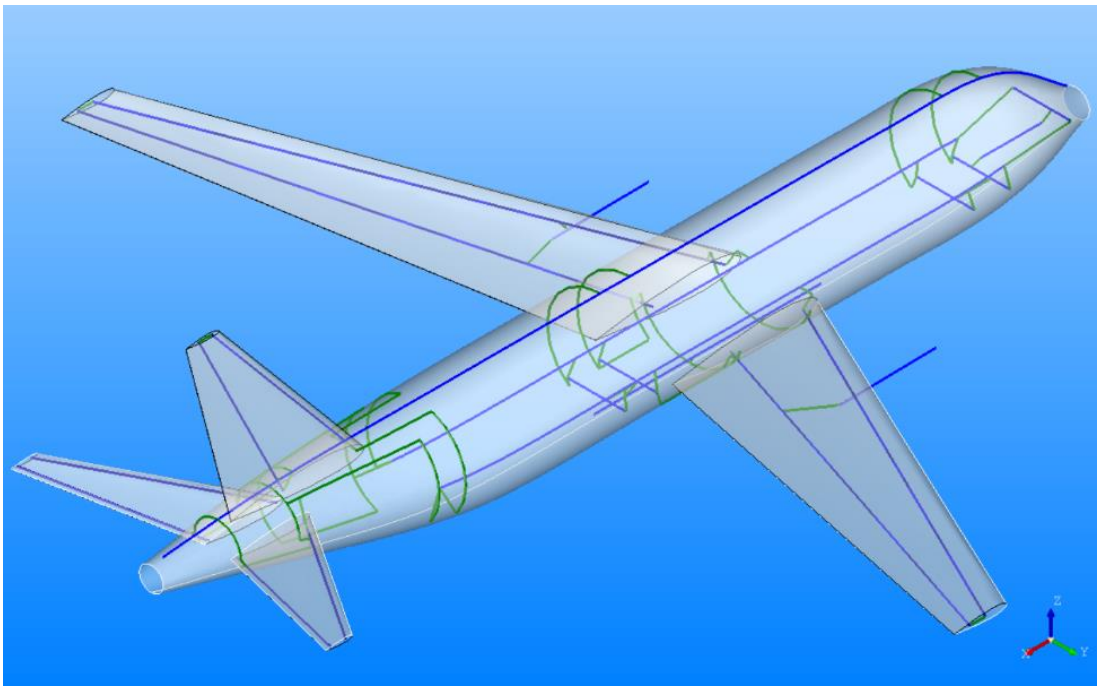

Figure 9: channel architecture (green and blue) for a conventional aircraft.

The AM has the option to use a low-fidelity structural model of the aircraft to create the network on which a channel architecture will be based and across which routing can be performed. The low-fidelity model of the structure could already be generated for wing-like structures using the *WingboxGenerator*, *MoveableGenerator* and *FlapGenerator*. With the addition of the *FuselageGenerator* the fuselage structure can also be included which enables the approximation of all relevant components at an early stage in the design process. Because the tool can be used early in the design process and is capable of rapidly generating alternative designs it is suitable for front-loading and design space exploration.

The *FuselageGenerator* is used to obtain frames and stringers by providing a *FuselageStationsLayout* based on a frame-pitch and a stringer-pitch. The resulting stringers and frames for the conventional aircraft example are illustrated in Figure 10. Additionally the AM also provides a *HeightwiseStation* for a fuselage floor definition with multiple *WidthwiseStations*. Note that in the *FuselageGenerator* these two stations have not been fully implemented but a preliminary implementation has been made specifically for this design case. This floor layout is used to

DOI: 10.13009/EUCASS2023-465

generate longitudinal and lateral floor beams as shown in Figure 11. Ultimately, these fuselage structural elements are combined with the wing-like structures into a single structural model on which the channel architecture is generated. The resulting channel architecture for the conventional aircraft example is shown in Figure 9 as well as Figure 12.

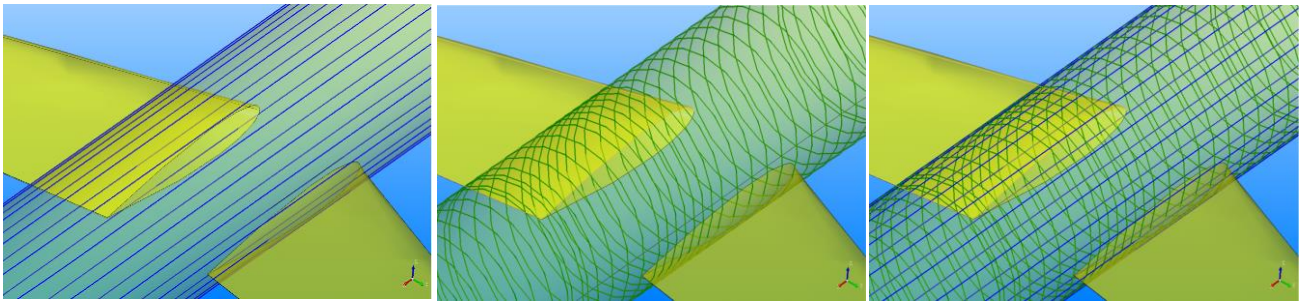

Figure 10: fuselage stringers (blue) and fuselage frames (green).

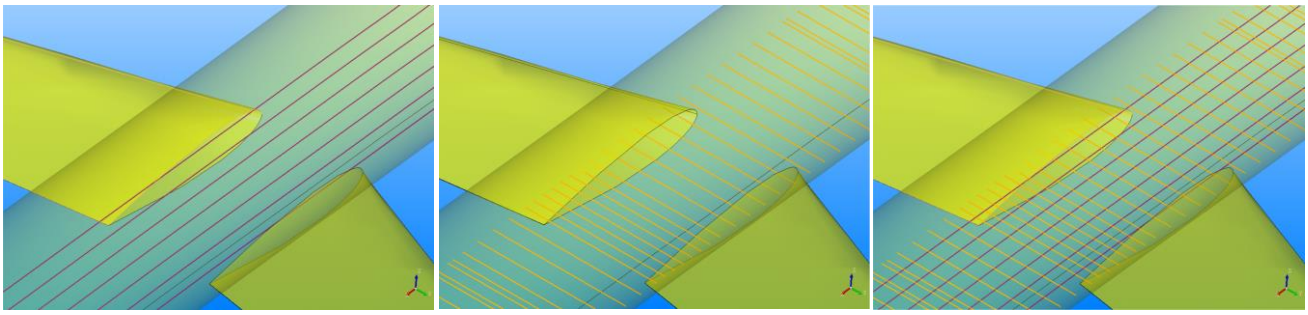

Figure 11: fuselage floor beams, both longitudinal (purple) and lateral (orange).

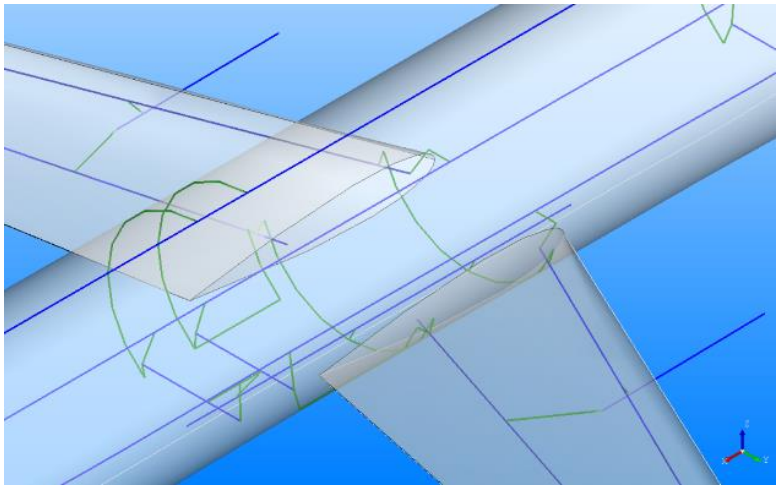

Figure 12: channel architecture (green and blue).

Mobility Fund: High-fidelity design case

In the design case used in the Mobility Fund project, the design of a small Urban Air Mobility (UAM) vehicle is considered. The final objective of this design case is to design and analyze the full tail section of the vehicle, which includes the fuselage tail section and empennage. For this, multiple products will be combined in a single generator object, as was explained in Section 4. The first step to this design case is to be able to model the rear fuselage section separately, before creating interfaces to the empennage.

The Outer Mold Line (OML) of the rear fuselage section is fixed and given as input to the design case by the customer. This input is given as a set of surfaces in STEP format. In figure 13, the visualization of the OML as used

DOI: 10.13009/EUCASS2023-465

in this design case is given. Since the area of interest is located at the rear of the fuselage, the shape is double curved and therefore more complex than a conventional center fuselage section.

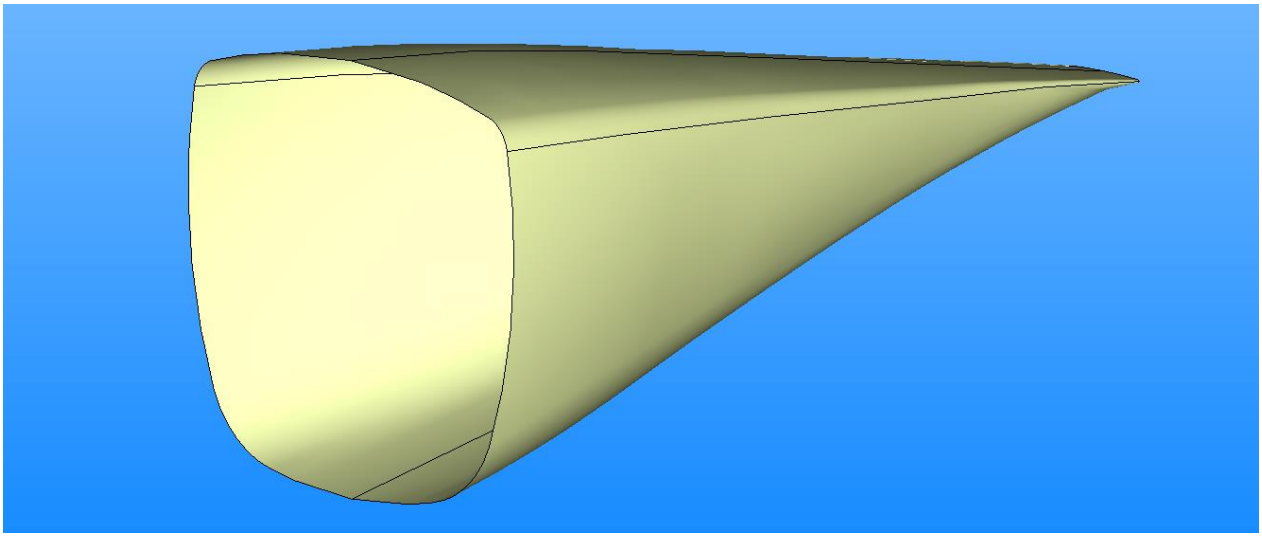

Figure 13: Reference geometry as used in the Mobility Fund design case.

At this stage, the fuselage consists of skin panels, frames and stringers. Using the *FuselageGenerator*, the structural layout of the fuselage can be configured in the tool input and the chosen configuration will be automatically applied to the structure. The configuration can be changed within seconds and different analyses can be applied to each configuration. In figure 14, an example of a configured fuselage structure is given. As can be seen in the figure, closed frames (bulkheads) are used in the structure, as well as L-shaped stringers along the circumference. The frames are positions on lengthwise fractions along the center line of the reference geometry. The stringers are places with a pitch along the circumferential fraction on the first and last frame stations. Because of the narrowing of the circumference along the center line, two stringers are stopped earlier.

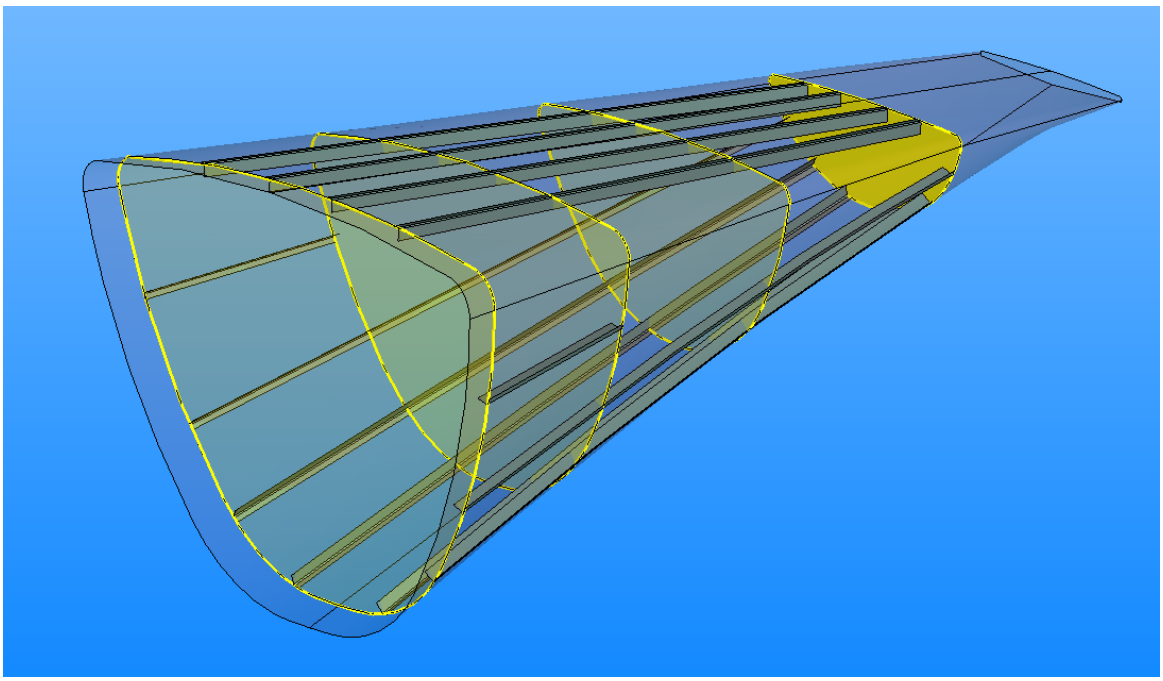

Figure 14: Reference geometry with skin panels, stringers and closed frames.
Some parts are transparent for visualization purposes.

At time of writing, analyses have not been implemented yet. It is the intention to make the fuselage modeler as versatile as the wing modeler currently is. This means that the same disciplinary models will be implemented for the fuselage as are currently available for wing-like structures, Because of the generic applicability of the stringer

DOI: 10.13009/EUCASS2023-465

primitives, the disciplinary models on primitive level are already present. The primitive disciplinary models for the frames will be added in a later stage, as well as the collector at the highest tree-level of the fuselage. Once the fuselage model is fully functional, the fuselage and wing box products will be used to create a full tail section product. This model will include the interfaces between wing and fuselage, as well as disciplinary models on product level (tail section), sub-product level (fuselage, wing box, etc.) and primitive level (spar, frame, stringer, etc.)

6. Conclusion

In this paper, the efforts made to extend the Multidisciplinary Modeler (MDM) package to enable the design and analysis of fuselage structures are described. This is done by firstly identifying differences and similarities in wing-like and fuselage-like structures. The next step was to identify which part of the package could be directly reused, which part of the package could be used with slight adaptions and which part should be developed completely new. It was concluded that the *Base* classes that are the foundation of every primitive could be completely reused. Furthermore, the classes that specify the product definition input could also be reused completely. Structural primitives that are present in a fuselage-like structure, but not in a wing-like structure were not yet part of the package. Therefore, these primitives are newly developed. In this paper, the example of a frame primitive is used to explain the process for such a fuselage specific primitive.

The new *FuselageGenerator* class is constructed similarly to the already existing generator classes, while implementing a new kind of structure. The same will be done to the structure of the product disciplinary models, which is currently still in development. Some of the next steps that will be taken are:

- Implementing new primitives to model a fuselage floor
- Configuration of the wing-box position, where interfaces to different structures can be defined
- Developing a fully integrated structure class, where a full tail section can be modeled

In the end, the addition of the fuselage structure to the MDM package enables a new range of trade studies that can be performed. Not only does this increase the speed at which GKN Aerospace can deliver engineering solutions to its customers, it also increases the Engineering quality by providing a standardized, automated approach to the design of aero structures.

Acknowledgement

The research presented in this paper has partially been performed in the framework of the DEFAINE (Design Exploration Framework based on AI for froNt-loaded Engineering) project and has partially been performed in the Dutch Mobility Fund project. The research has received funding from the *ITEA 3 programme* and the *Subsidieregeling R&D Mobiliteitssectioren (RDM)*.

References

[1] A.H. van der Laan, T. van den Berg, L. Hootsmans, “Integrated Multidisciplinary Engineering Solutions at Fokker Aerostructures”, 5th CEAS Air and Space Conference, Delft, 2015
[2] T. van den Berg, A.H. van der Laan, "A multidisciplinary modeling system for structural design applied to aircraft moveables", AIAA Aviation 2021 Forum, Virtual event, 2021.
[3] La Rocca, G. “Knowledge Based Engineering: Between AI and CAD. Review of a language based technology to support engineering design”, Advanced Engineering Informatics, vol. 26, no. 2, pp. 159-179., 2012.
[4] ParaPy, "The ParaPy platform" [online] URL: https://parapy.nl/features/ [retrieved 20th June 2023]
[5] T. van den Berg, H.S. van Manen, A.H. van der Laan, I. Ciobotia, D. Bansal, J. Sonneveld, “A multidisciplinary modeling system for aircraft structural components”, submitted to Joint 10th EUCASS-9th CEAS Conference, Lausanne, 2023
[6] A.H. van der Laan et. al., “Bringing Manufacturing into the MDO domain using MBSE”, AIAA Aviation 2022 Forum, Chicago, 2022
[7] J. Sonneveld, A.M.R.M. Bruggeman, G. La Rocca, T. van den Berg, H.S. van Manen, “Dynamic workflow generation applied to aircraft moveable architecture optimization”, submitted to Joint 10th EUCASS-9th CEAS Conference, Lausanne, 2023
[8] ZLIN Aircraft [online] URL: https://www.zlinaircraft.eu/en/ [retrieved 20th June 2023]

DOI: 10.13009/EUCASS2023-465

[9] M. Baan, et. al., "DEFAINE – Design Exploration Framework based on AI for front-loaded Engineering: Achievements and Open Challenges", submitted to Joint 10th EUCASS-9th CEAS Conference, Lausanne, 2023
[10] Samenvattingen Subsidieregeling R&D Mobiliteitssectoren (RDM) [online] URL: https://www.rvo.nl/subsidies-financiering/subsidieregeling-rd-mobiliteitssectoren/samenvattingen [retrieved 31 January 2023]