# Appendix A: Assignment description template

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Team n. 17 | Name | st. number | Name | st. number |
| Nelson Johnson | 4195787 | Şan Kılkış | 4192028 |
| Utilizing KBE to Frontload Preliminary UAV Design | | | | |
| Describe briefly your design case and make sure to address the following crucial questions:   * What is the design challenge your KBE app is supposed to tackle? * Why the use of KBE is supposed to be a good means to address such challenge? Thus, what are the characteristics of the problem at hand that match the strengths of KBE technology?   Imagine for a second that you work at a company specialized in producing light UAVs (1-20 kg MTOW) for multiple use-cases, and to be competitive, you desire agility and quick project proposals to cope with changing demands of the market. However, the challenge of exploring the design space and analyzing the feasibility of potential designs is a lengthy procedure as experienced from the DSE project. This process requires re-use of corporate knowledge from previous designs and application of repetitive statistical and empirical relations (rules) to aid in generation of preliminary models (geometry) which are then analyzed to obtain actual performance data. However, in the early stages of a design the overall geometry can change tremendously and without automation, the time it takes to analyze the design space will be a multiple of the time it takes for one preliminary design with the total number of designs to be analyzed. Thus, in a situation where time is limited (such as in the DSE or for project proposals) the depth to which these designs can be explored is constrained and often only qualitative information can be used to assess the strengths and weaknesses of the preliminary designs. Ultimately, without a streamlined process for this analysis, costly mistakes can be made due to the chosen design either not being feasible or not capable of fulfilling the customer’s requirements.    Figure 1: Illustration of How Conceptual and Preliminary Design Allocate Costs (La Rocca, 2018)  Current applications of KBE and KBSs are typically for detail design, yet most funds to a project are allocated during the preliminary design as illustrated by Figure 1. Thus, to be able to frontload this procedure, and free-up resources for more creative work, our KBE app will parametrically generate geometry based on output data from class I and II weight estimations, and requirements of different ranges, payload masses and use cases. At the moment, canard, conventional tail, and flying wing configurations will be potential options with the potential to add more later if time permits. The use cases include changes in the payload mass, payload size and other mission specific requirements such as hand launch-ability. The weight estimations are very repetitive rule based processes, which can take a lot of time in the conceptual design phase. Especially when an input such as payload mass changes, the entire class I and II weight estimations must be performed again. This KBE app will allow the generation of all the major components of a UAV as well as sub components such as the required battery, motor and propellor sizes. The battery and motor size will be based on a wing drag estimation from running AVL and an empirical estimation of fuselage and empennage drag. It will then generate a ‘.stl’ file of the final aircraft for input into CAD/CFD programs.  If the initial fixed-wing demonstrator is successful, due to the object oriented programming paradigm, this KBE app is open for future enhancement through adding analysis functionalities such as the ability to model alternative configurations such as multi-boom fuselages, BWB’s, and even multi-rotors. Finally, to present as a visual aid for the type of designs which the KBE will be valid for Figure 2a and 2b depict the upper and lower limits of maximum take-off weight respectively.       |  |  | | --- | --- | | Figure 2a: RQ-11 Raven (MTOW 1.9 kg) | Figure 2b: Penguin C (MTOW 22.5 kg) | | | | | |

|  |
| --- |
| Rule based parametric model requirements |
| Describe here the main functionalities of the rule based parametric model, which will be the core of your KBE application  The system components that will be present in the KBE app are summarized in a rough conceptual UML diagram presented below:   * What systems components/features will be included in the parametric model? * What are the main parameters used to define and control the various components/features? * What are the main (engineering) rules governing the definition/interface of the various model components? Identify the main sources you will use to capture knowledge * How will your app deal with rules violation? (warnings, automatic corrections, change suggestions)   **------------------------ max 2 pages including explanatory figures ------------------------------**  In the ParaPy GUI tree all main components of a typical UAV will be present, such as the wing, vertical and horizontal stabilizer, and fuselage along with all subcomponents such as parametrically selected/sized payload, propeller, motor, battery, and connecting rods. Also, if time permits shapes for the payload(s) can be generated based on reference modules such as cameras or synthetic aperture radars.  All of these component primitives will dynamically upscale to represent the change in requirements based on engineering rules encoded into their respective classes. Utilizing this approach, additional aircraft configurations will be possible to implement in the future since they will, for example, use the same wing, fuselage, and tail classes but orient them in different ways and alter the integration methods. An example is the fuselage primitive which will utilize a parametric center section which scales up or down with the size of the payload and battery. To then allow flexibility in configurations and mounting options, different nose and tail shapes can be further added to this class. For example an extended nose section can be used to mount canards or a blended shape with tangency conditions can be used to create an aerodynamic fairing between the truncated center fuselage and a propeller.  The parameters which size the wing area are from the class I weight estimation, while those for the fuselage and tailplane area (tail volume) being from the class II weight estimation. Furthermore, the majority of knowledge that will be used was either presented during the bachelor, obtained from literature reports and reference UAV’s, or generated during the DSE for a project on a modular UAV for surveillance. The utilization for this knowledge will be distributed within the parent classes of each primitive. As is the case with KBE systems, the inference mechanism and rules are scattered and hard to centralize. Thus each individual primitive will encompass its own set of sizing rules and weight estimations. An overview of some of these rules is provided below:  **Global Aircraft**  The global aircraft class (root class) will have the responsibility of translating the mission requirements from the excel sheet to top-level design parameters such as:   * Maximum Take-off Weight * Wing Loading * Thrust Loading * Maximum Lift Coefficient * Wing Span, Aspect Ratio, Taper Ratio (Assumed) * Target Drag Coefficient → Target L/D   These top level design parameters will be derived from either best-guesses, as in the case of wing span, aspect ratio, taper ratio, and target L/D, or from empirical relations such as MTOW estimation. Once these values are derived the wing loading and power loading diagrams can be constructed. These will limit the design space to an allowable region. Furthermore, the capture of practical rules such as maximum throwable mass/size of the aircraft or an emphasis on endurance will lead to the selection of the design point within the plausible design space. For example in the case where the requirement of hand-launch ability is the design driver, the design point will be chosen such as to minimize wing-loading and select a higher than normal value of trust loading. On the other hand if endurance is the driving requirement then a high wing-loading will be chosen to allow a smaller wing-surface and thus less drag.  This class will contain user-inputs for configuration selection as well as integration options to enable the end user to override the “best-guesses” that the KBE app provides. Finally, attributes with lazy evaluation will also allow the user to ask for updated performance values from AVL, masses, and c.g. locations.  **Wing (Conventional and Canard)**  Once top level design parameters are defined by the global aircraft class, a wing object can be instantiated based on once again initial guesses from best engineering practice for the airfoil, sweep (if necessary), and dihedral, and taper ratio. If triggered by another class or the end-user, an attribute containing an AVL simulation will return, for this initial guess, the current performance parameters of the wing. Following this, the end-user can either make changes manually through the input or call on a optimizer attribute to cycle through an airfoil database or incrementally vary a wing geometry parameter. For example the flying wing will require sweep, a reflexed airfoil, and some dihedral angle assumed from reference aircraft. After positioning all components (battery, engine, avionics, fuselage), the sweep must then be iteratively chosen by the KBE app for longitudinal stability.  **Horizontal Tails (Conventional and Canard)**  After the class II weight estimation, the required tail volume will be found. Using an empirical relation for an adequate Sh/S will allow for the calculation of the HT Area and arm. More empirical relations will be used for the taper ratio and span. Finally, another AVL simulation can be triggered via attribute(s) to find performance data of the constructed tail such as the Cm\_alpha contribution.  **Vertical Tails (Conventional and Canard)**  As this is not the current emphasis of our KBE app, we will add a simple empirical relation to size the VT. Further, the VT will be parameterized in a similar fashion to the wing and horizontal stabilizer.  **Fuselage**  The center section of the fuselage will be the same for all configurations. It will follow a reference length/diameter ratio with the maximum diameter is determined from the required width from the battery and payload. The ends of the fuselage will vary, depending on the engine location. On the side with the engine, there will be a conical fairing allowing for a smooth change in diameter and on the other, there will be a hemisphere, or other fairing shape terminating the fuselage.  We will need to add a rule for a minimum fuselage length in the case that the required volume from the battery and payload don’t drive it. In the case that the required fuselage length is very large (due to battery and payload), there must also be a warning supplied to the user stating  **Wing-Fuselage Integration**  We are not yet sure how to do this, but our current idea is to apply a fillet with radius as a function of the local chord length between the wings and fuselage.  In order to not break the parametric model of ParaPy a comprehensive value checking algorithm must be placed. These will trigger warnings or error messages if certain values are deemed outside of a suitable range. For example if the payload mass supplied by the user is a negative value. Furthermore, whenever best engineering practice is violated the user might be alerted by a warning message. An example would be a warning to reduce a high value of sweep for a conventional configuration since it is not needed for the flight regime. |
| Internal analysis capabilities |
| * What analysis modules will be implemented **inside** your KBE application, thus coded in ParaPy. At least one internal analysis module should be present.   **------------------------ max half page ------------------------------**  This app will take mission reqs and transform them into initial design parameters for the sizing of all component parts in the aircraft model.  The Class I weight estimation must choose a feasible design from the generated wing and thrust loading diagrams. This choice will be influenced by the input choice of hand-launchability.  Within the KBE app, the center of gravity as well as surface area for various components (wings, fuselage, payloads, batteries, avionics, engines) will be calculated using the built in ParaPy attributes. These are then used to estimate the C.G. location when combined with component masses from Class II weight estimation and for drag estimation based on component surface areas.  There will be an analysis of geometry conflicts between parts. For example, when the wing is attached to the fuselage, it cannot intersect internal components such as batteries, avionics or payload. |
| Link(s) with external analysis module(s) |
| * What **external** analysis module will be connected to your KBE app? How will your app interact with such applications? At least one external analysis tool/module should be present.   **------------------------ max half page ------------------------------**  AVL will be used to calculate the wing drag at cruise conditions. This will then be added to an empirical relation for the fuselage and empennage drag to derive the required thrust power. This will be used to size the battery (and indirectly the fuselage). Furthermore, AVL results can be called within a while-loop slowly varying a single parameter such as wing sweep. The resultant data can then be used to optimize that parameter. |

|  |
| --- |
| Input data handling capabilities |
| * What data sets will be provided as input to your KBE app? * In which format will the input data sets be defined? * Which data (sub)set will be interactively editable in the ParaPy GUI?   **------------------------ max half page ------------------------------**  An excel file including mission requirements on Range/Endurance, Payload Mass and dimensions and Configuration choice will be the tools input file. The user will be asked if they want to have it hand launchable and this will influence the choice of the design point in the Class I estimation.  A set of airfoil ‘.dat’ files will be supplied to the KBE app. A set of reflexed airfoils will be used for the flying wing configurations, while normal airfoils for the conventional and canard main wings. Symmetric sections will be used for the VT and HTs for the conventional wings, while a cambered section used for the canard surface.  Within the Parapy GUI, the range/endurance, payload mass and dimensions, configuration type, airfoil choice and hand-launchability will all be editable.  Near the payload dimension input, there will be a suggestion to respect a minimum length/width ratio to fit inside a realistic fuselage. Within parapy, the payload density will be calculated and if it is relatively low, or the length/diameter is too low, an error with another suggestion will be displayed.  For now, we will assume a propulsive efficiency but if time permits, we would like to increase this by adding a optimal propellor choice algorithm based on the flight speed and optimal engine RPM. |
| Output data reporting capabilities |
| What output files is the KBE app supposed to generate and in what format?  At least one STEP(or IGES) file and one output file containing results from the analysis modules.  **------------------------ max half page ------------------------------**  The KBE app will output a ‘.stl’ file with the aircraft ready for CAD/CFD import. Also, a data file with the relevant aircraft dimensions will be output. Another data file with the wing aerodynamic analysis results will be output.  If possible, we would also like to allow the user to save their current aircraft design (after making changes in GUI) to be able to open it and continue another time. |