31st Congress of the International Council of the Aeronautical Sciences
Belo Horizonte, Brazil; September 09-14, 2018

ON-DEMAND SMALL UAS ARCHITECTURE SELECTION AND RAPID MANUFACTURING USING A MODEL-BASED SYSTEMS ENGINEERING APPROACH

Cedric Justin*, Arun Ramamurthy*, Nathan Beals, Eric Spero**, Dimitri Mavris***
***Georgia Institute of Technology, **United States Army Research Laboratory**

Keywords: *Systems Engineering, On-Demand Design, Manufacturing*

Abstract

Small Unmanned Aircraft Systems (UAS) have become ubiquitous over the past decade. This is in part due to the convergence of many new technologies relevant for UAS applications. This offers opportunities for new architectures and new designs of UAS not previously feasible. As engineering and acquisition organizations focus on small UAS as possible solutions to bridge capability gaps identified in future operational environments, a need has emerged to better understand the relationships between requirements, system architectures, and technologies. An integrated and executable system analysis capability featuring requirement analysis, automated design, as well as rapid prototyping and manufacturing through additive manufacturing, has been developed. It relies on an executable framework based on a model-based systems engineering approach to the design of small UAS. A use case is demonstrated with the design and manufacture of a small UAS satisfying the need of expeditionary forces for rapid development of tailor-made capabilities.

1 Introduction

The past decade has seen a dramatic increase in the number of small Unmanned Aircraft Systems (UAS) being used for recreational, commercial, and military applications. The number of applications suitable for small UAS operations runs a gamut never previously anticipated, from automated package delivery to remote sensing and weapon delivery. According to recent estimates from the Federal Aviation Administration [1], there are currently more than 750,000 small UAS operating in the National Airspace System. Focusing more closely on U.S. Army small UAS operations, these systems are deployed mostly to support tactical operations through the collection of intelligence, surveillance, and reconnaissance information. Owing to the time-sensitivity of these operations, these systems would be best deployed on-demand to acquire intelligence in real time. In fact, the U.S. Army UAS roadmap for 2010-2035 [2] recommends that UAS be used to enable decentralized decision-making stating that:

> *"UAS require and enable accelerated multi-echelon, decentralized decision-making, and execution, significantly changing the tempo and dynamics of operations. Lower echelon leadership must be empowered with authority and bandwidth to employ UAS as their changing situation dictates, operating at a tempo that is faster than higher echelon leadership can affect."*

However, the logistics involved with procuring small UAS means that latencies are introduced, which can adversely impact the gathering of intelligence 'on-demand'. A paradigm shift is needed so that small UAS can be designed, deployed, and operated at the lower echelon of leadership to ensure an appropriate level of reactivity. In this context, an 'on-demand' solution is a solution for which an asset appropriate for the mission is made available within a reasonable time-frame. Because equipping users with small UAS assets able to meet unforeseen needs poses design and logistical challenges, 'on-demand' solutions also require on-demand requirement analysis, 'on-demand' design, and 'on-demand'

manufacturing. Consequently, to ensure that the needs of users are properly identified and met, the following capabilities need to be developed:

- A process to elicit and analyze customer requirements
- A process to generate designs in close to real-time
- A process usable by customers unfamiliar with the design of small UAS
- A manufacturing process usable near the point-of-need to limit transportation delays
- A visualization interface to convey the need for the relaxation of requirements when requirements are conflicting

Within the U.S. Army aviation community, engineering and acquisition organizations are beginning to focus on small, hand-held, portable UAS as materiel solutions to capability gaps envisioned in future complex operational environments. What is lacking however is an understanding of the relationships between requirements, UAS architectures, and technologies which can support the assessment of these systems in future environments. An integrated system analysis capability based on analysis, prototyping, and experimentation can provide a hands-on way for a wide range of decision-makers to gain a quantitative understanding of how design decisions drive and are driven by the technologies on-hand.

Leveraging existing efforts in the fields of automated design and rapid manufacturing, this research aims at providing a framework bringing together requirements analysis, automated design using model–based systems engineering techniques, visualization highlighting the capabilities of various UAS architectures, rapid prototyping to verify and validate requirements, and finally, rapid manufacturing to deploy assets on-demand.

1.1 Literature Review

The capability to perform automated design given a set of requirements has been previously investigated. Cheng et al. [3] study the use of component models to generate feasible multicopter designs using a model-based systems engineering approach. Fisher et al. [4] investigate the use of a modular design process in order to generate a family of multicopter and fixed-wing vehicle designs meeting specific mission requirements. Fisher et al. [5] and Locascio et al. [6] further refine the modular design process using model-based systems engineering techniques to automate the design process.

One enabler of an ‘on-demand’ solution is the ability to perform in-situ prototyping and manufacturing. New manufacturing techniques such as additive manufacturing, enable both the rapid manufacturing of unique systems and the forward-deployment at the point-of-need. In turn, this enables users to rapidly create prototype solutions to address unforeseen or unanticipated problems. Additive manufacturing applied to rapid prototyping and manufacturing has been previously investigated by Mangum et al. [7] for small multicopters using a simple hover-based energy and endurance analysis.

While research has been carried out to develop model-based representations of small UAS and to automate design and manufacturing processes, few previous efforts have focused on understanding the capabilities of various UAS architectures and the impact of future technologies on these capabilities.

1.2 Problem Formulation

Owing to the modularity of small UAS, investigating the capabilities offered by different architectures requires a flexible framework in which design and configuration changes can be traced and evaluated quickly. Established approaches to the design of complex systems typically involve multiple analysis tools where documents describing the system being analyzed are passed around. Given the modularity of UAV systems, this approach can be slow and tedious owing to the necessity for manual configuration of the numerous documents involved. The nature of the documents as well as their possible lack of readability may limit transparency, thus hindering the ability of designers to immediately assess when and where parameters of interest are changed.

To address this possible lack of transparency during design, researchers [4] [5] [8] [9] have proposed a model-centric approach which ensures that a set of *‘models’*, defined by and visible to the designer, dictates the flow of information during the design process. An

inherent feature of the model-centric approach is the ability to create models at multiple levels of abstractions and have these models interact with each other. This multi-layered abstraction approach enables quick replacements of modular entities, thereby enabling the investigation of *'variants'* of existing designs [5]. The following subsections detail the implementation of an executable model-centric framework and its application to design space exploration and probabilistic analyses of various UAS concepts.

1.3 Description of Application

Throughout this research, a single set of requirements defining a UAV mission is used to illustrate the proposed approach. The customer is an expeditionary warrior who describes the intended reconnaissance mission using natural language as highlighted in Table 1. The UAV is launched during a covert mission and retrieved later at a home base.

Table 1: Use Case Mission Description

Phase	Description
Manufacture	The UAV shall be manufactured within 48 hours.
Take-off	The UAV shall be hand launched and shall clear a 10 m high forest canopy within 50 m.
Cruise Climb	The UAV shall climb to a height of 100 m above ground level at a rate of 1.5 m/s within 600 m.
Cruise	The UAV shall fly a distance of 1,800 m within 2 min.
Loiter	The UAV shall loiter over the point of interest for at least 5 minutes.
Cruise	The UAV shall fly a distance of 2,700 m to a retrieval point within 3 min.
Landing	The UAV shall land within 15 m on a turf field without damage.
All	The UAV shall weigh at most 5 kg and shall carry a payload of 0.15 kg

2 Model-Based Systems Engineering Approach

A Model-Based Systems Engineering (MBSE) approach [10] builds on the model-centric view established by Model-Based Engineering (MBE) [11] and extends it to the realm of systems engineering. The model-centric view of MBE uses models as an integral part of establishing a baseline. The MBSE approach relies on the development of a baseline made of elements necessary to the life-cycle management of a product. This includes activities such as requirements analysis, performance analysis, design optimization, as well as requirements verification and validation. Thus, the MBSE framework accounts for all the elements of the life-cycle of the product, from the beginning of the conceptual design phase with the definition and handling of requirements, to the detailed design and manufacturing, and to the later life-cycle phases of validation, operations support, and maintenance. Overall, an MBSE framework improves communication across development teams, reduces design cycle times, and reduces risks through the identifications of failures earlier in the design cycle [12].

2.1.1 SysML and Object-Oriented Systems Engineering Methodology

While MBSE provides a framework to perform systems engineering-oriented design, a language and a methodology are needed to enable the development of a product life-cycle model. A comprehensive survey of methodologies for the application of MBSE is provided by Estefan [13]. A review of modeling languages suitable for MBSE implementation is performed by Reilley [14]. Based on their findings, the Systems Modeling Language (SysML) is selected for this research in order to model small UAS as it provides a modeling language through the extension of a subset of Unified Modeling Language (UML) protocols. The four key pillars of SysML are:

- The structure elements and diagrams, which describe the components making up the system and provides a definition of permitted interactions between these components
- The behavior element and diagrams, which dictates the functional and behavioral aspects of the system
- The requirements, which captures the desired characteristics in terms of system behavior, design, and operation
- The parametrics, which defines relationships and bindings between the different attributes of the structure package

2.1.2 Towards an executable SysML-based MBSE Framework

SysML is not inherently executable which forces the utilization of external applications and model converters to perform the various activities pertaining to the design process [5] [14]. Moreover, design engineers may not always

be familiar with the fundamentals of systems engineering, which means that the adaptation to the new paradigm of SysML-enabled MBSE can be challenging. Given the popularity of the Python programming language within the engineering community, a light-weight executable modeling framework is developed in Python to implement the MBSE-enabled design and manufacturing of small UAS. This choice of programming language as well as the architecting of the framework are influenced by past research efforts [15]. The new implementation makes nevertheless substantial modifications over previous work.

First, owing to the nature of engineering design, the framework merges the behaviors and parametrics into a single *'process'*. The process, formulated as a design structure matrix (DSM) [16], defines both behavioral and evaluation analyses similarly to the approach adopted in the structure elements and diagrams. DSMs are the accepted standard for the representation of processes in the engineering community [17].

Next, elements of the system are represented using an abstract data structure named a *'tree'*. Interactions between tree elements are defined by a new type of standard attribute called *'interface'* added at each node in the tree. This attribute dictates the way one node interacts and interfaces with another. User-defined functions describing interactions are allowed to ensure flexibility in the definition and representation of the interface.

This modified data structure, the implementation of design processes as DSMs, and the executable nature of the framework, enable the rapid re-creation of traditional design processes, within the model-centric view, without the need for intermediate model converters. This enables designers to bypass the learning curve associated with traditional SysML practices during the development of models.

3 Methodology

3.1.1 Modeling UAS

The modeling of UAS per the specifications of MBSE requires the development of computational representations of the requirements, the system structure, the component interfaces, and the processes associated with the design and manufacturing activities. The following sections detail the development of models necessary to analyze the capabilities of different UAS architectures, to evaluate their suitability for specific missions, and to study the propagation of technological and technical uncertainties to system-level metrics of interest.

3.1.2 Requirements Modeling

The traditional Forsberg and Mooz Systems Engineering 'Vee' model [18] relies on having independent processes for the decomposition of requirements, and their verification during manufacturing and testing. While requirements verification and validation are essential elements in the automation of design processes, the decoupled approach to the performance verification present numerous issues such as the need for fabrication prior to the start of the verification, the lack of transparency in the requirements flow and during decisions, and an increase in design cycle time and cost. The presence of an executable MBSE framework, in which requirements are directly mapped to metrics of interest, has been identified as a means to mitigate the challenges associated with this decoupled approach. Indeed, when models are updated, analyses associated with these updated models are automatically triggered and yield updated metrics of interest indicating whether requirements are satisfied.

For the development of the requirements, two additional prerequisites are imposed:

- Requirements are stated in natural language, which means they need to be translated from natural to engineering language using appropriate metrics.
- The framework needs to be capable of establishing relationships between elements of the model and stated requirements.

The elicitation of requirements often leads to a description of the vehicle's desired functionality which can be used to derive the vehicle's target performance measures. For instance, users may specify requirements in terms of the mission profile which are then converted into engineering metrics for the design process. To achieve these goals, natural language processing (NLP) [19] and text parsing are used. The process used to translate mission attributes

into verifiable requirements is illustrated in Fig. 1 with the mission highlighted in Table 1 and using the cruise segment description:

"The UAV shall fly a distance of 1,800 m within 2 min."

First, a dependency tree [20] is built to identify the elements of the sentence, thereby breaking complex requirements into sets of simpler requirements. In the above example, there are two objects of prepositions. Thus, the cruise requirement can be decomposed into two requirements: one deals with the range associated with the attribute '*distance*' and corresponding value '*1800 m*', while the other deals with the duration associated with an implicit attribute '*time*' and corresponding value '*2 min*'.

Second, once a requirement is decomposed into its constituents, a vector embedding [21] is generated for each individual constituent. This vector represents the projection of the natural language requirement into an n-dimensional real space where similar requirements occupy a common region of the space. This is exploited to identify the nature of the requirement using a pre-trained classifier. This classification algorithm provides a description of the nature of the requirement which is then passed downstream to subsequent steps.

Third, a parameter and value identification algorithm is used to identify numerical parameters. A dependency tree is generated to identify the 'nummod' tags [22] which corresponds to values associated with parameter modifiers. For instance, the numbers 1,800 and 2 are identified as the values of interest, and for each, the association of a parameter is attempted. Both the parameters 'distance' and 'time' are identified via a mapping of the objects of prepositions associated with their numerical values to a predetermined set of parameters. The algorithm relies on the use of a set of pre-specified mappings to identify appropriate parameters for unspecified requirement parameters. The nature of the requirement serves as an input to this algorithm to handle special cases where requirement parameters are not cardinal numbers. This is for instance the case for the vehicle launch mechanism (hand launch) or the type of landing surface (turf).

Finally, the relationships that exist in the given description are used to identify the source node. This process of identification of the source node enables the extraction of the validation relationship for the requirement. In the above example, the relationship extraction algorithm identifies that for both requirements, distance and time, the nominal subject is the noun 'UAV'. This noun should match one node in the structures package whose name or description matches the target identified by the relationship identification algorithm. Corresponding parameters (distance and time) or their mappings (distance mapped to range) on that node can then be populated accordingly.

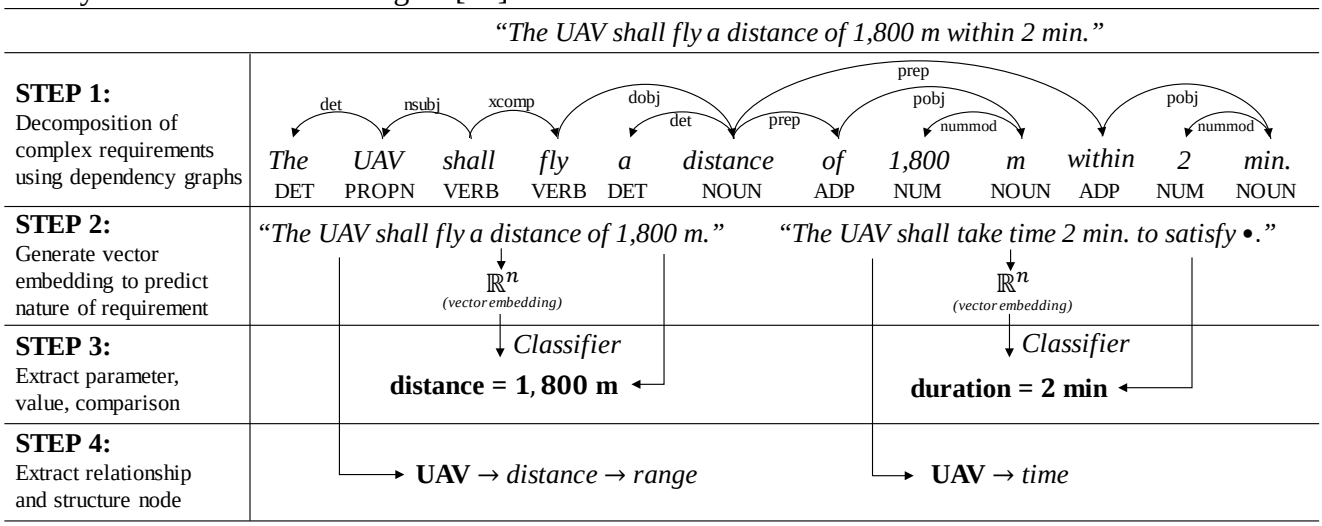

Fig. 1. Algorithm for the processing of requirements using Natural Language Processing

With this information, the requirement tree is ready to be populated with the new subtree constructed from the specified description. The leaf nodes of this subtree are assigned a unique semantic ID and have the attributes of description, parameter, value, relationship and source.

To verify the satisfaction of requirements, a two-stage approach is taken. First, if a requirement has an associated source, parameter, relationship, and value, then the evaluated value of the parameter of the source node is checked to see if the requirement is satisfied. The requirement is satisfied, if and only if, its children are evaluable and satisfied. Upon satisfaction, the evaluation proceeds to a sibling requirement node. This is highlighted in Fig. 2 for the take-off requirement. This recursive approach is employed to identify how requirements flow from the leaves to the root of the tree, therefore providing users with a clear picture of the source of infeasibility, if any.

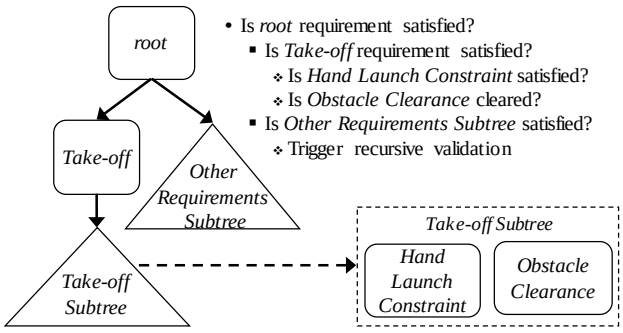

Fig. 2. Process flow for the validation of the requirements tree

The diagrams in Fig. 3 show the SysML equivalent of the generated requirement based on the description of the cruise requirement of Table 1. The current natural language processing capability is highly tuned to the problem under consideration since the lexicon (corpora) used to train the algorithm is specific to the small UAS application under consideration.

Fig. 4 on the following page details the requirements breakdown that serves to guide the evaluation of any design generated. The requirements specification is comprised of the mission requirement (mission profile, system-level requirements such as weight, payload etc.), and the manufacturing requirement (manufacturing time and cost).

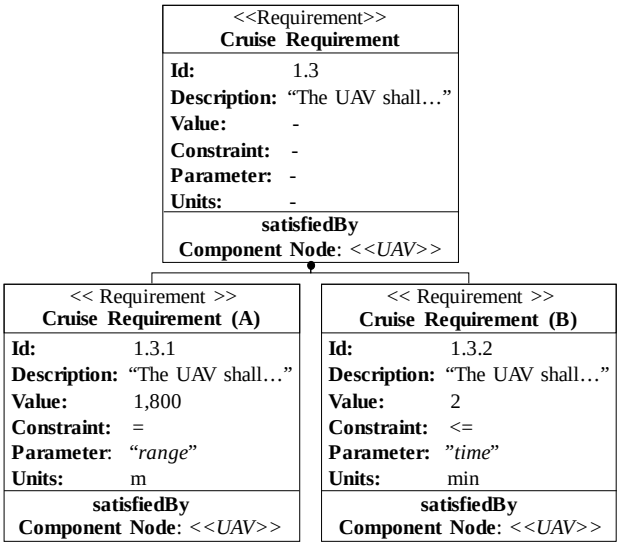

Fig. 3. SysML-like representation for the generated cruise requirement

3.1.3 Structure Modeling

One aim of the research is to contrast different architectures of UAS meeting a set of 'must-have' requirements. Consequently, a quick evaluation of the different variants is necessary and metrics related to their overall design, capabilities, and manufacturing performance are used for benchmarking purposes.

Each design variant is defined as a combination of a set of compatible components and a set of values for the scalable design parameters. The purpose of decomposing the vehicle is not only to enable the transition from one design variant to another by replacement of one modular component with another, but also to establish the interface specifications for these components. The methodology used in the modeling of structural elements of the system is similar to the one established by Fisher [5]. A functional and physical decomposition of the vehicle is carried out to identify the nature of the vehicle and the set of possible alternatives for the components of the system. In parallel, an enumeration of the manufacturing alternatives is performed to identify the machines that can be used to manufacture the product. These decompositions form the first layer of the *'structure tree'* shown in Fig. 5 and comprised of two children: the design and the manufacturing. The design child is further decomposed into the component systems and architecture alternatives.

Requirement 1: ***Mission Requirement*** • The UAV shall complete successfully the mission as per the mission description.	*Requirement 1.1:* ***Take-off requirement*** • The UAV shall be hand-launched and shall clear the 10 m high forest canopy within 50 m.	*Requirement 1.1.1:* ***Take-off requirement (A)*** • The UAV shall be hand-launched.
		Requirement 1.1.2: ***Take-off requirement (B)*** • The UAV shall clear an obstacle of 10 m.
		Requirement 1.1.3: ***Take-off requirement (C)*** • The UAV shall satisfy the climb requirement within 50 m.
	Requirement 1.2: ***Climb requirement*** • The UAV shall climb to a height of 100 m above ground level at a rate of 1.5 m/s within 600 m.	*Requirement 1.2.1:* ***Climb requirement (A)*** • The UAV shall climb to a height of 100 m above ground level.
		Requirement 1.2.2: ***Climb requirement (B)*** • The UAV shall climb at a rate of at least 1.5 m/s.
		Requirement 1.2.3: ***Climb requirement (C)*** • The UAV shall satisfy the climb requirement within 600 m.
	Requirement 1.3: ***Cruise requirement*** • The UAV shall fly a distance of 1,800 m within 2 min.	*Requirement 1.3.1:* ***Cruise requirement (A)*** • The UAV shall fly a distance of 1,800 m.
		Requirement 1.3.2: ***Cruise requirement (B)*** • The UAV shall satisfy the cruise requirement within 2 min.
	Requirement 1.4: ***Loiter requirement*** • The UAV shall loiter over the point of interest for at least 5 min.	
	Requirement 1.5: ***Cruise requirement*** • The UAV shall fly a distance of 2,700 m to a retrieval point within 3 min.	*Requirement 1.5.1:* ***Cruise requirement (A)*** • The UAV shall fly a distance of 2,700 m.
		Requirement 1.5.2: ***Cruise requirement (B)*** • The UAV shall satisfy the cruise requirement within 3 min.
	Requirement 1.6: ***Landing requirement*** • The UAV shall land within 15 m on a turf field and without damage.	*Requirement 1.6.1:* ***Landing requirement (A)*** • The UAV shall land within a distance of 15 m on a turf field.
		Requirement 1.6.2: ***Landing requirement (B)*** • The UAV shall not be damaged during landing.
	Requirement 1.7: ***Payload requirement*** • The UAV shall carry a payload of 0.15 kg.	
	Requirement 1.8: ***Weight requirement*** • The gross weight of the UAV shall not exceed 5kg.	
Requirement 2: ***Manufacturing Requirement*** • The UAV shall be manufactured within 48 hours.	*Requirement 2.1:* ***Manufacturing requirement*** • The manufacturing time for the UAV shall not exceed 48 hours.	

Fig. 4. Requirements breakdown driving the design process for the UAV

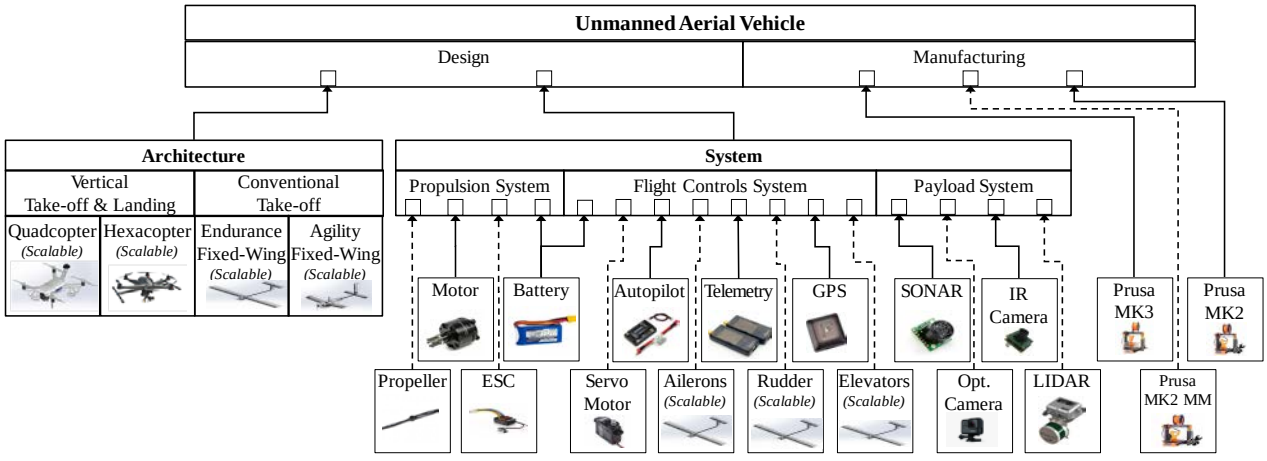

Fig. 5. Coupled functional and physical decomposition of the UAV

The architecture subtree is then decomposed into alternatives, namely the fixed-wing and the multicopter platforms. Each of these platforms is further decomposed into two specific product families, namely agility-focused and endurance-focused families for the fixed wing platform and quadcopter and hexacopter for the multicopter platform.

The components subtree is functionally decomposed into the constituent subsystems: the propulsion subsystem, the flight-control subsystem, and the payload subsystem. These are further physically decomposed to identify the constituent components. The propulsion system is decomposed into motor, battery, electronic speed controller, and propeller. The flight-control subsystem is decomposed into servo motor, electronic speed controller, battery, and control surfaces. Since the flight-control system and the propulsion system share power from the battery, an interface relation is established for these subsystems to enforce the shared relation between these components. An additional interface is defined at the multicopter level which dictates the multiplicity for a set of components. Similar exercises are undertaken for the payload subtree in order to generate the structure tree.

While this decomposition process defines completely the architectural and design space, it does not yet account for interactions between the different components. These interactions can be modeled as physics-based or compatibility-based constraints established through the interface mechanism. For example, a certain motor may require a specific nominal voltage. An interface constraint restricting the selection of batteries unable to provide this voltage can be established for this motor, thereby reducing the set of possible variants. One final extension made to the structure tree is the indication that multiple payloads can be selected. Indeed, it is assumed by default that each of the leaf node children of a given subtree is incompatible with its siblings, (if a vehicle has a battery of a given type, then it cannot have a battery of another type as well). Such behavior can nonetheless be changed by overriding the compatibility interface between the components or by specifying the possibility of multiple selections at any subtree.

Having decomposed the entire system, key attributes of components are gathered and classified into two groups. The first contains commercially-off-the-shelf alternatives while the second contains specifically designed parts. Databases containing specifications are established for the commercial off-the-shelf alternatives such as batteries, motors, propellers, electronic speed controllers and payloads. These configurable databases are then used to populate the leaf nodes for the appropriate component node. The final step in the decomposition is the identification of the design parameters for the components classified as being specifically designed. These, in addition to the design parameters at the architectural level, dictate the set of scalable parameters that are to be considered during the design process. In contrast to the methodology established by Fisher [4], mappings between scalable parameters are not defined within the structure package. Instead, they are handled by the processes package and the associated design structure matrix.

Fig. 6 provides a representation of one of the component sets providing the SysML-like representation for the motor alternatives (only two amongst many motors are shown). This approach uses a separate section in the block diagram for the interfaces assigned to the component.

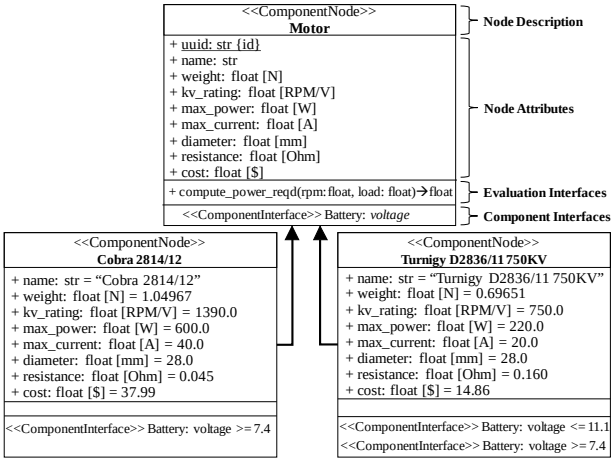

Fig. 6. SysML-like representation of a motor and its alternatives

While the definition of the structure tree and the interfaces within are created manually, these need to be defined only once for each system under study. Indeed, when a component instance is created, it is stored in the database at which

point it can be called upon to recreate the element in the structure tree during subsequent use. Modifications to already created components are possible and result in the creation of a separate instance of the component node in the database. The ability to modify existing components to create new component instances can significantly reduce the design cycle time and the workload when modeling complex systems.

3.1.4 Parametrics Modeling

Traditionally, the function of parametrics in SysML is the modeling of physics-based relations defining the behavior of elements of the system. While such models are applicable to the design of a system during the conceptual and preliminary phases, detailed design of a system often requires discipline-centric (or even multi-disciplinary) analyses in which multiple components and their interactions are considered. Given the fact that analyses within each discipline can be carried out at different levels of fidelity, an extension to the parametrics is needed. In order to simplify the computational representation of the design workflow, a tree data structure is retained. This is a result of the multi-layered abstraction of the design process. This multi-layered abstraction or hierarchy is divided into three levels:

- Workflows, which define the evaluation sequence of nested entities and the data flow between them
- Disciplines, which indicate the disciplinary analysis that can be performed
- Analyses, which represent the set of alternatives within any disciplinary analysis

In order to represent the processes involved during engineering design, an attempt is made to mimic an established standard for their representation using design structure matrices. Design structure matrices provide a representation of the flow of information between disciplines and are visually appealing for convoluted workflows with numerous parameters passed around.

The workflow element of the framework can be viewed as a container which is associated with an algorithm. A library of algorithms is provided such that a variety of numerical computations can be performed, such as design of experiments, numerical optimization, and uncertainty quantification. This library feature is exploited to perform both the capability exploration and the uncertainty quantification. Prior to defining the characteristics of the workflow, it is nevertheless necessary to define the actual process that the algorithm will operate on. The workflow can host a set of interconnected disciplines, a set of nested workflows, or a combination of both. The discipline functions as a container for a set of analysis owing to its multi-fidelity nature. For a discipline to be evaluated, there has to be exactly one analysis that is active at any instant of time. If a discipline does not need to be evaluated, logic can be included to deselect any constituent analysis at which point the execution of the discipline is skipped until a selection is turned back on again. The final layer of the hierarchy is the analysis node. The analysis represents the actual computation that is performed in the workflow. These, in addition to the workflows, represent the executable components of the framework.

The executable nature is achieved by the registration of an evaluation source for every evaluation node which are represented by the created workflows and analyses. In the case of the workflow, the evaluation source would be an algorithm, while in the case of the analysis the evaluation source would represent some numerical computation. These evaluation nodes are then linked to the evaluation ports, which are abstract methods definitions on the structure. This linkage definition ensures that attributes of the structural node are visible to the evaluation source. Optional interface specifications, in the form of value bindings, can be specified for the evaluation sources. The bindings efficiently map parameters from the evaluation to and from the registered structure node as an evaluation is undertaken. These values bindings can also map the nodes themselves in cases where multiple parameters have to be accessed, thus reducing the amount of coding necessary. Interfaces across disciplines can be defined at the workflow level such that mapping between parameter values are efficiently handled upon completion of the execution of a discipline. These parameter mappings can be defined through equations that

dictate the *'interface'* between scalable parameters in the case of the UAS application.

Using this framework, the design and manufacturing process associated with UAS are represented as workflows consisting of multiple disciplines. The contents of the workflow differ depending on the case considered.

3.1.5 Vehicle Sizing and Synthesis Workflow

The various candidate architectures retained require different types of analyses in order to check their suitability to fulfill the mission requirements and complete the mission.

Fixed-wing sizing and synthesis

The fixed-wing vehicle family is built around two modular designs shown in Fig. 7. One design is optimized for longer endurance and range missions, while the other design is optimized for agility-based urban missions. These designs are modular with a tailor-made structure housing the payload as well as other off-the-shelf components. The tailor-made structure is composed of a fuselage section, as well as a wing, vertical tail, and horizontal tail which are sized for the mission.

Fig. 7. Endurance-focused and agility-focused fixed-wing baselines

The objective of the sizing and synthesis analysis is to determine the appropriate wing loading and the optimum size of the wing and empennage. The vertical and horizontal stabilizers of the empennage are sized using the tail-volume coefficient method. Wing area is therefore the only parameter driving the size of the vehicle. The wing loading is determined by carrying out Mattingly's constraint analysis [23] which relates power-to-weight and wing loading. The analysis is adapted for small vehicles operating at low-Reynolds numbers. With the constraint analysis, typical performance requirements such as take-off constraint, obstacle clearance constraint, climb performance constraint, cruise speed constraint, service ceiling constraint, and landing constraint can be accounted for. Several new features are also implemented to recognize constraints specific to UAVs as well as constraints related to manufacturing and the use of off-the-shelf components.

One constraint is related to the ability to specify how the UAV is launched. If the vehicle is ground-launched, then take-off roll and obstacle clearance constraints are implemented. If the vehicle is hand-launched, then a maximum stall speed constraint is implemented to account for the ability of the user to manually launch the vehicle.

A manufacturing constraint is also included to represent the time available to produce a mission-ready UAV. The manufacturing time is directly linked to the wing size of the vehicle owing to the need to produce more and larger ribs when the wing size increases. To be included in a constraint diagram, this manufactured constraint must be expressed in terms of wing loading. First, the structure mass, denoted m_{str} and defined as the total mass m_t minus the off-the-shelf electronic component mass m_c, is regressed against wing area *S* using experimental data. The regression is highlighted in (1) for both agility (A) and endurance (E) focused vehicles using wing areas ranging from 0.05 m^2 to 0.70 m^2.

$$m_{str} = m_0 + m_1 \cdot S \quad (1)$$

$$\begin{cases} (m_0, m_1)_A = (0.298\ kg, 2.629\ kg \cdot m^{-2}) \\ (m_0, m_1)_E = (0.145\ kg, 2.034\ kg \cdot m^{-2}) \end{cases}$$

In turn, this regression allows the derivation of a relationship (2) between wing loading *W/S* and wing area accounting for the mass of off-the shelf components and the standard gravity *g*.

$$S = \frac{g \cdot (m_c + m_0)}{\frac{W}{S} - g \cdot m_1} \quad (2)$$

Using experimental data, a new regression is performed to estimate manufacturing time as a function of wing surface and thus wing loading. This is displayed in (3).

$$t = t_0 + t_1 \cdot \frac{g \cdot (m_c + m_0)}{\frac{W}{S} - g \cdot m_1} \quad (3)$$

$$\begin{cases} (t_0, t_1)_A = (19.1\ hr, 123.9\ hr \cdot m^{-2}) \\ (t_0, t_1)_E = (15.5\ hr, 100.3\ hr \cdot m^{-2}) \end{cases}$$

This means that a manufacturing time constraint can be expressed in the constraint diagram with a vertical line given in (4)

$$\frac{W}{S} = g \cdot m_1 + \frac{t_1 \cdot g \cdot (m_c + m_0)}{t - t_0} \quad (4)$$

Once all relevant mission and manufacturing constraints are plotted, the existence of a design space may be revealed. If a design space exists as highlighted in Fig. 8, there will be an area representing feasible combinations of power-to-weight and wing loading in-between constraints. With the choice of battery, propeller, and motor, the available power-to-weight is represented using a new '*component selection*' constraint. Available power is indeed determined by the selection of components, and all feasible designs are found along a single component selection curve representing the available power-to-weight ratio for various wing loadings and thus various wing areas. The equation corresponding to this curve is given in (5).

$$\frac{P}{W} = \frac{P}{g \cdot \left[m_c + m_0 + m_1 \cdot \frac{g \cdot (m_c + m_0)}{\frac{W}{S} - g \cdot m_1} \right]} \quad (5)$$

With a given wing-loading selection, the wing area is calculated using the structure weight and the off-the-shelf component weights. The take-off, landing, best endurance, and best range speeds can then be estimated.

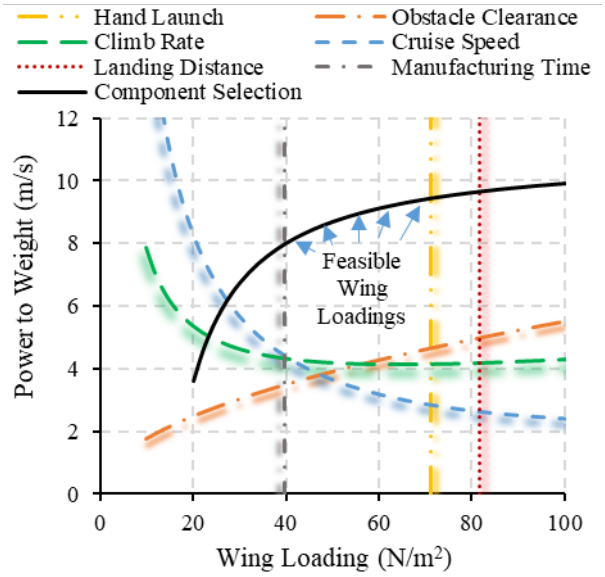

Fig. 8. Constraint diagram (Four 28 W motors and one 3-cell 5,800 mAh battery)

Multicopter performance estimation

The multicopter family is built around two modular designs represented in Fig. 9. One design is a quadcopter, while the other is a hexacopter. Both designs use various off-the-shelf electronic components that are fitted on a tailor-made structure. The structure is built with a central hub hosting most of the electronics and payload, and four or six arms housing the electronic speed controllers, the motors, the propellers, and the multicopter landing skids.

Fig. 9. Quadcopter and hexacopter

The sizing of the multicopter is performed using a power-based and energy-based method. Experimental wind-tunnel data [24] for the lift and drag coefficients of the multicopter structure in forward flight is used to create surrogate models of the lift and drag coefficients at different pitch angles γ. These surrogate models are used during the trim analysis to estimate the multicopter attitude (α_{TPP}, γ) at any given speed V_{inf} and climb rate V_c as described in Fig. 10. In turn, this enables the estimation of the required thrust using a simple point-mass balance of forces.

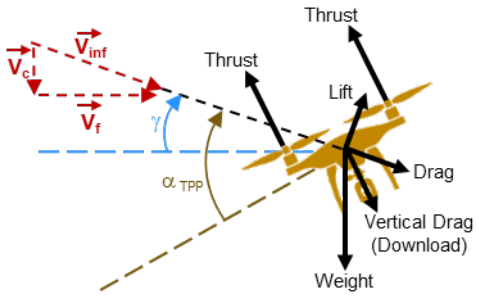

Fig. 10. Balanced of forces for multicopter

The required thrust is used next to estimate the rotor angular speed, the induced velocity, and the required power as highlighted in Fig. 11. This is carried out using blade element momentum modeling corrected for low Reynolds numbers [25]. The induced velocity is needed to estimate the download on the multicopter structure. As a result, an iterative procedure is required to converge on pitch angle, thrust, rotor angular speed, and required power.

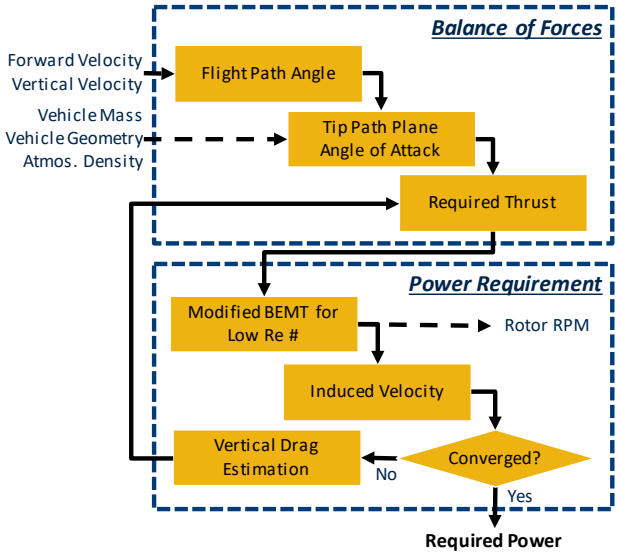

Fig. 11. Multicopter power requirement

With the ability to estimate power in any flight condition, the required power is calculated for each mission leg and each maneuver specified by the user. The best endurance and best range speeds are estimated for missions that do not specify any speed. In order to create a multicopter design, the motor, propeller, and battery combinations are exhaustively explored using the process highlighted in Fig. 12.

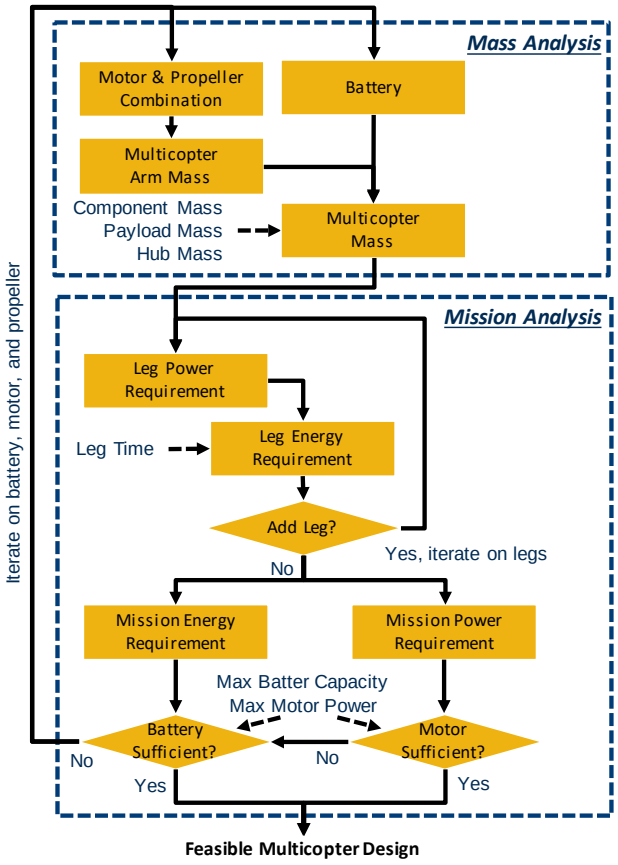

Fig. 12. Feasible multicopter design search

The multicopter arm length is determined by the propeller diameter to ensure sufficient clearance around the propellers. Consequently, the manufacturing time, which depends only on the multicopter arm length and choice of manufacturing machine, is essentially driven by the propeller diameter selection.

3.1.6 Use Cases

The MBSE approach is exercised on two use cases: a concept capability exploration use case and an uncertainty propagation use case.

Case 1: Concept Capability Exploration

For the concept capability exploration, the corresponding workflow is articulated around four main steps highlighted in Fig. 13.

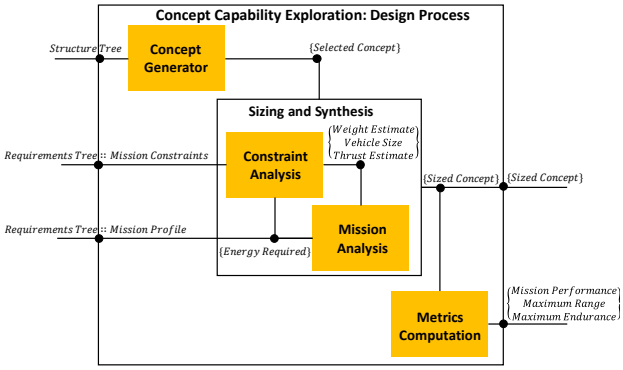

Fig. 13. UAV design process workflow for the capability exploration case

The first step is a *'concept generator'* and its function is to generate a design platform from the structure tree that adheres to the defined structural interface specifications. Having generated the concept, a *'sizing and synthesis'* workflow is utilized to analyze the performance of the vehicle. The sizing and synthesis workflow is comprised of the *'constraint analysis'* and the *'mission analysis'* and it iteratively sizes the vehicle by altering the scalable parameters in the structure tree while ensuring that mission requirements are met. During the process of sizing, validation of mission requirements is performed until a feasible design is found, if one exists. After having produced a design, its performance is evaluated using metrics of interest in the *'metrics computation discipline'* to estimate any additional capability over the minimum performance requirement stated by the user. This additional capability typically results

from the choice of off-the-shelf components that may exceed the minimum required capability.

Fig. 14 illustrates the manufacturing process workflow for the concept capability exploration case. The manufacturing process starts upon completion of the design process when a feasible design is identified. In such a scenario, the manufacturing process starts with the *'machine selector'*, whose role is to select the set of machines available. The machine selector module ensures that a machine is compatible with the designed product and associated manufacturing process. Once a compatible machine is selected, the *'model generator'* is triggered in order to create a computer-aided design (CAD) model of the design with the scalable parameter and components updated according to the outcome of the design process. The CAD model is exported next to a format usable by additive manufacturing or laser-cutting machines. A *'process simulator'* discipline is then launched in order to estimate the manufacturing time to create the various tailor-made scalable parts. Finally, a *'metrics computation'* block is triggered to compute the total cost of the product. The cost is estimated by summing the price of the off-the-shelf components and by adding a manufacturing cost representing the cost of producing the different parts (material and machine amortization).

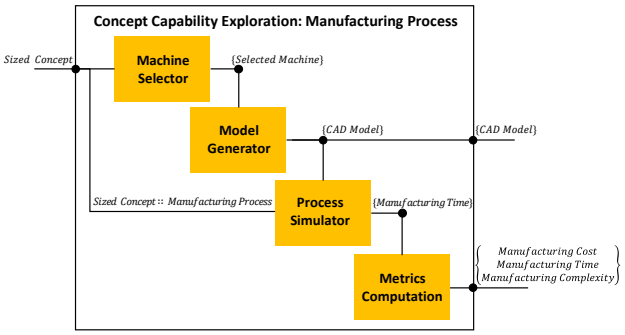

Fig. 14. Workflow representing the manufacturing process analysis for the UAV

Case 2: Uncertainty Propagation

For the uncertainty propagation case, a design is assumed to be preselected and the behavior of various metrics of interest is studied as technical and technological uncertainties are introduced. The workflow remains identical to that of the concept capability exploration case, except that the concept generator discipline is removed. This workflow is illustrated in Fig. 15. The workflow for the manufacturing process remains identical to the concept capability exploration case.

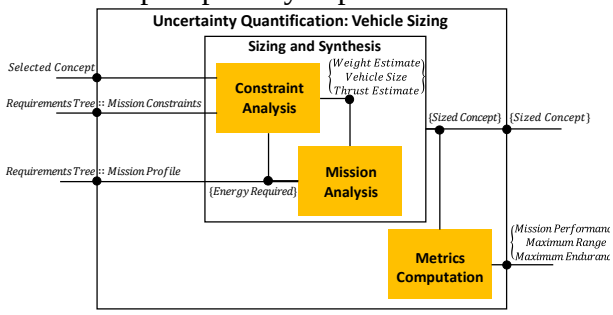

Fig. 15. Workflow for uncertainty quantification during design process

3.2 Analysis

This section documents the analysis setup used to investigate the two use cases of interest. Table 2 summarizes the set of parameters identified as being key metrics of interest being tracked over the course of the analyses.

Table 2: Key metrics used to evaluate the various UAV concepts generated

Design Parameters	Manufacturing Parameters
Mission performance	Manufacturing Time
Maximum Range	Material Cost
Maximum Endurance	Total Cost

For the first use case, a screening design of experiments is performed to analyze the concept capabilities. For the second use case, Monte Carlo simulations are implemented to estimate the impact of technological uncertainties related to battery technology (specific energy density) and material characteristics (thickness of additively manufactured parts). The uncertainty in battery technology is attributed to the improvements in battery specific energy density over time. An annual growth of 5% is assumed [26] [27] and the analysis is performed to assess the capability of various UAV architectures around the 2030 timeframe. Similarly, the uncertainty in material characteristics represents future improvements in additive manufacturing. The availability of better manufacturing materials and better manufacturing techniques will enable a reduction of the factor-of-safety associated with the thickness of the various parts designed. As a result, the thicknesses of shelled and beam components are probabilistically varied.

The uncertain battery specific energy improvement is represented using a truncated normal distribution (truncated at 10% below and above the mean) with a coefficient of variation of 5%. The mean value accounts for the expected improvement over time from the current average specific energy density of 138 Wh/kg to 228 Wh/kg by 2030, with upper and lower limits set at 202 Wh/kg and 247 Wh/kg respectively. The uncertain improvements in material characteristics is modeled using a uniformly distributed factor-of-safety ranging from 0.7 to 1.0 and representing the relative thickness of additively manufactured components (current baseline value set at 1.0)

3.2.1 Visualization and results

The following paragraphs summarize some of the results generated from the analyses. The top graph of Fig. 16 plots the manufacturing time against the maximum endurance for feasible designs. One salient result is that only endurance-focused fixed-wing and quadcopter vehicles can meet the 48 hours manufacturing time constraint. While fixed-wing vehicles generally outperform multicopters in terms of endurance and range, these additional capabilities come at the cost of generally longer manufacturing times. Unless there is a clear need for vertical take-off and landing, a fixed-wing vehicle seems to be the preferred solution when the mission length is unknown at the time of launch given that these vehicles offer more range and more endurance at similar weights.

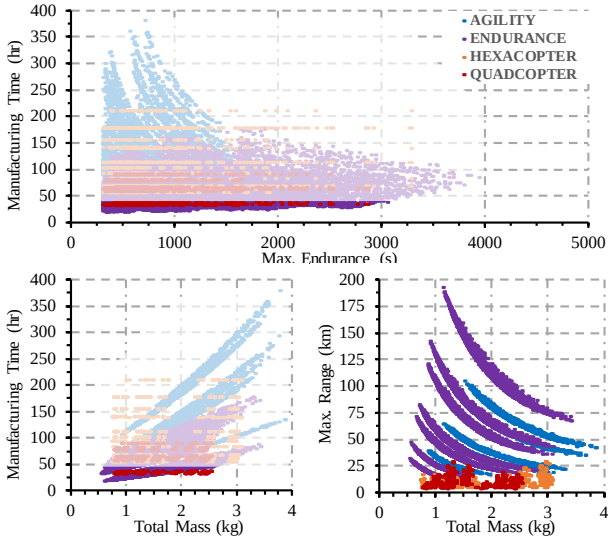

Fig. 16. UAS capabilities

Fig. 17 highlights the impact of incorporating technological uncertainties on two product variants, namely the endurance-focused fixed-wing and the quadcopter. The figure indicates that the sensitivity to the material and manufacturing uncertainties are comparatively greater for the fixed-wing vehicle. The sensitivity to the battery specific energy is greater for the quadcopter.

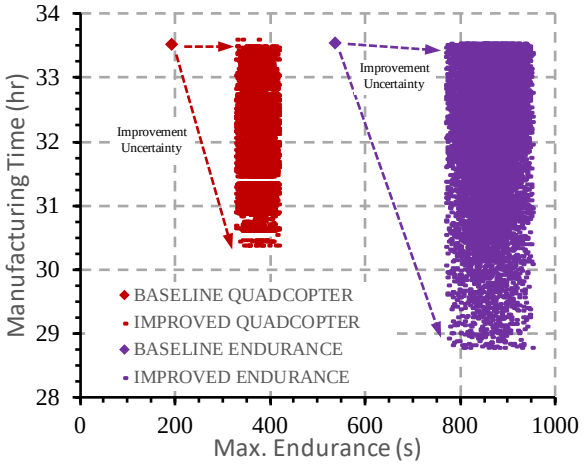

Fig. 17. Impact of technology uncertainties on manufacturing time and endurance

4 Conclusion and future work

An executable environment for the on-demand design and manufacture of small UAS has been developed. This environment relies on the elicitation of required mission and performance capabilities. Using natural language processing, these capabilities are translated into engineering requirements. A sizing and synthesis environment for small UAS is used next to design fixed-wing and multicopter vehicles satisfying these requirements. Real-time tracking of requirements enables the user to identify if and when requirements cannot be made, and whether these requirements should be relaxed. Computer-aided design models are automatically created and sent to manufacturing machines for production and assembly. In addition, the environment enables the investigation of the capabilities of small UAS architectures, as well as the study of the impact of technological uncertainties on small UAS capabilities.

Future work will include the investigation of new architectures such as a hybrid fixed-wing vehicle featuring rotors for vertical take-offs and landings.

References

[1] Federal Aviation Administration, "You & UAS," FAA Safety Briefing, Washington, DC, 2017.

[2] U.S. Army, "U.S. Army Roadmap for Unmanned Aerial Systems 2010-2035," U.S. Army, Ft Rucker, AL, 2010.

[3] A. Cheng, Z. Fisher, R. Gautier, K. D. Cooksey, D. N. Mavris and N. Beals, "A Model-Based Approach to the Automated Design of Micro-Autonomous Multirotor Vehicle Systems," in *AHS Forum 72*, West Palm Beach, FL, 2016.

[4] Z. C. Fisher, D. Locascio, K. D. Cooksey, D. N. Mavris and E. Spero, "ADAPT DESIGN: A methodology for enabling modular design for mission specific SUAS," in *42nd Design Automation Conference*, Charlotte, NC, 2016.

[5] Z. Fisher, K. D. Cooksey and D. N. Mavris, "A model-based systems engineering approach to design automation of SUAS," in *2017 IEEE Aerospace Conference*, Big Sky, MT, 2017.

[6] D. Locascio, C. Ramee, K. D. Cooksey K and D. Mavris, "A Model-Based Approach to the Automated Design of Small Unmanned Airplanes," in *AIAA AVIATION 2015*, Washington, DC, 2015.

[7] P. Mangum, Z. Fisher, K. D. Cooksey, D. Mavris, E. Spero and J. Gerdes, "An Automated Approach to the Design of Small Aerial Systems Using Rapid Manufacturing," in *ASME Design COnference*, Boston, MA, 2015.

[8] J. Humann and E. Spero, "Modeling and simulation of multi-UAV, multi-operator surveillance systems," in *2018 Annual IEEE International Systems Conference (SysCon)*, Vancouver, BC, Canada, 2018.

[9] L. Petnga, "Constraint-driven Design Specification for Small Unmanned Aircraft Systems," in *2018 AIAA Aerospace Sciences Meeting*, Kissimmee, Florida, 2018.

[10] INCOSE, Systems Engineering Handbook, INCOSE, 2007.

[11] NDIA Systems Engineering Division M&S Committee , Final Report of the Model Based Engineering (MBE) Subcommittee, NDIA, 2011.

[12] L. E. Hart, "Introduction To Model-Based System Engineering (MBSE) and SysML," 30 July 2015. [Online]. Available: www.incose.org/docs/default-source/delaware-valley/mbse-overview-incose-30-july-2015.pdf.

[13] J. A. Estefan, "Survey of Model-Based Systems Engineering (MBSE) Methodologies," *INCOSE*, 2008.

[14] K. A. Reilly, S. J. Edwards, R. S. Peak and D. N. Mavris, "Methodologies for Modeling and Simulation in Model-Based Systems Engineering Tools," in *AIAA SPACE 2016*, Long Bech, CA, 2016.

[15] S. Balerstrini-Robinson, D. F. Freeman and D. C. Browne, "An object-oriented and executable SysML framework for rapid model development," in *2015 Conference on Systems Engineering Research, Procedia Computer Science 44*, 2015.

[16] A. A. Yassine, "An Introduction to Modeling and Analyzing Complex Product Development Processes Using the Design Structure Matrix (DSM) Method," University of Illinois at Urbana-Champaign, 2004.

[17] T. Browning, "Applying the design structure matrix to system decomposition and integration problems: a review and new directions," *IEEE Transactions on Engineering Management, Vol: 48, Issue: 3*, pp. 292 - 306, 2001.

[18] K. Forsberg and H. Mooz, "The Relationship of System Engineering to the Project Cycle," in *The 12th INTERNET World Congress on Project Management*, Oslo, Norway, 1994.

[19] C. D. Manning and H. Schütze, Foundations of Statistical Natural Language Processing, Cambridge, Massachusetts: The MIT Press, 1999.

[20] M.-C. de Marneffe and C. D. Manning, "Stanford typed dependencies manual," Stanford University, 2008, rev. 2016.

[21] T. Mikolov, I. Sutskever, K. Chen, G. Corrado and J. Dean, "Distributed representations of words and phrases and their compositionality," *Neural Information Processing Systems*, vol. 26, p. 3111–3119, 2013.

[22] "nummod: numeric modifier," 30 June 2018. [Online]. Available: http://universaldependencies.org/en/dep/nummod.html.

[23] J. D. Mattingly, W. H. Heiser and D. T. Pratt, Aircraft Engine Design, Reston, VA: AIAA, 2002.

[24] N. Willard, "Multirotor Aerodynamics ," Georgia Institute of Technology, 05 2016. [Online]. Available: https://www.rcgroups.com/forums/showatt.php?attachmentid=8993682&d=1463328343. [Accessed 01 07 2018].

[25] M. H. McCrink and J. W. Gregory, "Blade Element Momentum Modeling of Low-Re Small UAS Electric Propulsion Systems," in *33rd AIAA Applied Aerodynamics Conference*, Dallas, TX, 2015.

[26] J. Janek and W. G. Zeier, "A solid future for battery development," *Nature Energy*, vol. 1, 2016.

[27] C.-X. Zu and H. Li, "Thermodynamic analysis on energy densities of batteries," *Energy Environ. Sci*, vol. 4, no. 8, p. 2614–2624, 2011.

[28] M.-C. de Marneffe and C. D. Manning, "Stanford typed dependencies manual," http://nlp.stanford.edu/downloads/lex-parser.shtml, 2008.

Contact Author Email Address

For information concerning this research, please contact Cedric Justin:
email: cedric.justin@gatech.edu

Copyright Statement