Proceedings of 2018 AIAA/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference, AIAA SciTech Forum, 8-12 January 2018, Kissimmee, Florida, USA,
Paper number AIAA 2018-1067
DOI:10.2514/6.2018-1067

Towards Automating the Sizing Process in Conceptual (Airframe) Systems Architecting

Yogesh Bile[1], Atif Riaz[2], Marin D. Guenov[3] and Arturo Molina-Cristóbal[2]
School of Aerospace, Transport, and Manufacturing
Cranfield University, Cranfield, Bedfordshire, MK43 0AL, United Kingdom

Presented is a method for automated sizing of airframe systems, ultimately aiming to enable an efficient and interactive systems architecture evaluation process. The method takes as input the logical view of the system architecture. A source-sink approach combined with a Design Structure Matrix (DSM) sequencing algorithm is used to orchestrate the sequence of the sub-system sizing tasks. Bipartite graphs and a maximum matching algorithm are utilized to identify and construct the computational sizing workflows. A recursive algorithm, based on fundamental dimensions of additive physical quantities (e.g., weight, power, etc.) is employed to aggregate variables at the system level. The evaluation, based on representative test cases confirmed the correctness of the proposed method. The results also showed that the proposed approach overcomes certain limitations of existing methods and looks very promising as an initial systems architectural design enabler.

I. Introduction

THE conceptual phase is a crucial part of the systems design process. It can generally be broken down into architectural design synthesis (architecting) and parametric design synthesis (sizing). In general, during the architecting process, new architectures are created by infusing new technologies into existing architectures, referred to as baselines. The baseline can be modified by adding or deleting new components and connections. During the modification process, the architect wishes to swiftly find the effect of the architectural changes on the system level performance. This involves sizing of the modified sub-systems and then obtaining the system level performance. Depending on the evaluation results, the architect may decide on further modifications. The above labor and knowledge-intensive process can be repeated several times as a result of frequent modifications during the conceptual design stage. In this regard, the motivation behind the presented in this paper-work has been the improvement of the effectiveness and the efficiency of the early stage systems architecting process. It is assumed that the architecting takes place in four domains: Requirement, Functional, Logical and Physical (RFLP)[1]. In a recent work, Guenov et al.[2] specified the interface between the functional and logical views. Their approach is extended here to include sizing and is restricted to the systems requirements and logical representations as the input to the systems sizing process.

Existing automated sizing methods[3–7] (albeit utilizing manual tasks) have been developed for specific system/sub-system architectures, which limits their applicability. The more generic (multidisciplinary optimization) methods[8,9] still need a human specialist to formulate the sizing problem. In this context, the aim of the research presented here has been to automate the sizing process wherever possible, thus making the systems architecting process more interactive.

The rest of paper is organized as follows: the next section summarizes the terminology and gives more details on the state of the art related to the presented research. The proposed approach is described in Section III. The evaluation of the proposed method is presented in Section IV. Finally, conclusions are drawn and future work is outlined in Section V.

[1] Research Student, Centre for Aeronautics, Cranfield, MK43 0AL, United Kingdom.
[2] Research Fellow, Centre for Aeronautics, Cranfield, MK43 0AL, United Kingdom.
[3] Professor, Head of the Centre for Aeronautics, Cranfield, MK43 0AL, United Kingdom, AIAA Senior Member.

Published by AIAA. This is the Author Accepted Manuscript issued with:
Creative Commons Attribution Non-Commercial License (CC:BY:NC 4.0).
The final published version (version of record) is available online at DOI:10.2514/6.2018-1067
Please refer to any applicable publisher terms of use.

II. Background

This section begins with the definition of several terms used in the description of the proposed approach, followed by a summary of the state of the art.

A. Definitions

Function: Functions are intended behaviors of a component or a (sub) system. During the functional analysis process, the top level system function is decomposed into lower level functions. Functions which cannot be decomposed further are called leaf-functions.

System: A system is a set of interrelated components working together toward some common objective or function. The system can be represented as a pyramid of hierarchy[10] as shown in Figure 1. In this paper, the terms 'system' and 'product' are used interchangeably. Examples of a system are aircraft, car, satellite, ship etc.

Sub-system: A sub-system is a collection of components fulfilling a higher level function of the system. A sub-system is a recursive aggregate, i.e., can be comprised of other subsystems and so on.

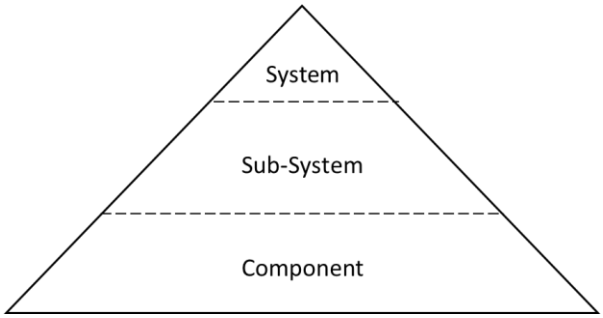


Figure 1. Pyramid of the system hierarchy.

Component: A component is a single element of the systems logical view, fulfilling a leaf-function.

Computational model: This is the computer code of the mathematical models describing the physics (behavior) and characteristics of a system, sub-system or component. A model can have single or multiple inputs and outputs. Depending on details that are exposed to the user, a model can be a 'white box' or 'black box'. The black-box model exposes only inputs and outputs, but no internal details; the white box model exposes both. An example of a black-box model is shown in Figure 2(a), where v1 and v2 are the default input variables and v3 and v4 are the outputs.

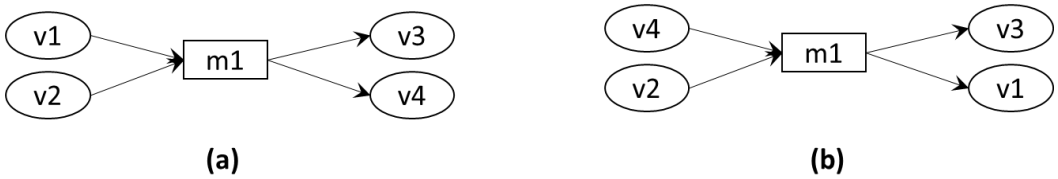


Figure 2. Black-box computational model and reversed computational model.

Reversed model: A model where one or more of its default input variables are designated as outputs and vice versa (Figure 2(b)). The reversal of a model is implemented through the application of numerical methods (e.g. optimization methods such as Newton's method[11–13]).

Computational workflow: A network of computational models. It shows the flow of information among the models. The workflow may contain reversed models.

Strongly connected component (SCC): A subset of the workflow forming a circuit (or cycle). The SCCs are solved using numerical methods such as the Newton-Raphson method[11] and Fixed Point Iteration[12–14].

B. State of the art

As stated in the introduction, there are existing automated sizing methods which have been developed for specific domain architectures. This, however, limits their applicability, while the more generic methods still need a human specialist to formulate the sizing problem.

Chakraborty et al.[3,5] developed an aircraft-specific conceptual design method, integrating sub-system architecture sizing with aircraft and propulsion sub-system sizing. It was further improved to cater for novel more electric architectures[4,15]. The method is aircraft specific while the computational workflows for sizing individual sub-systems as well as calculating system level performance are manually created. These manual activities would be required for updating the workflow whenever architectural changes are made. Similarly, Liscouet-Hanke[6,7] exploited a functional analysis for the design of aircraft power systems. Here, the aircraft level sizing models and power system sizing models are coupled during the sizing process, depending on the aircraft parameters that are required for sizing the systems. Similar to Chakraborty's method, Liscouet-Hanke's method is domain specific and also involves manual activities to construct the computational workflows during the sizing of the power system.

Generic multidisciplinary design methods such as Analytical Target Cascading (ATC)[8,16,17], Collaborative Optimization (CO)[9] and Multi-disciplinary Optimization (MDO)[18] can be used to design engineering systems. However, these methods require a specialist to formulate the design problem for a given system architecture. Here, the specialist needs to manually translate the architectural information into the design problem formulation.

General (transient) system simulation tools can also be used for sizing. For example, the (de facto) standard SysML4Modelica[19] was developed to establish a link between the architecture description (SysML) and Modelica[20]. Similarly, Wang and Dagli transformed the SysML diagrams to an executable model represented as Colored Petri Nets[21,22]. However, these approaches require "trial and error" alteration of model parameters' values (which remain fixed during a simulation run) to get desired model-variable values. The values of the model variables are changed during a simulation run. This "trial and error" of parameter values can be performed manually, or alternatively, a numerical optimization problem needs to be set by a human specialist. Similar problems occur with other physics-based modelling languages such as EcosimPro/EL[23,24] and Bond Graph[25].

During the sizing process, a sequencing of the design activities needs to be devised. The Design Structure Matrix (DSM) [26–28] is a well-known and extensively used system engineering method for managing inter-related entities. In the present research, the DSM has been employed for sequencing sizing tasks. Graph-based techniques can also be used for the composition/orchestration of computational workflows. Serrano[29] used bipartite graphs to model constraints network (of equations) and bipartite matching, based on linear programming to sequence the equations for their execution. This method was applied by Rudolph and Bolling[30] for the conceptual design of airship. The method is applicable to equations, i.e., models with only one output and multiple inputs. Similar approaches are described in Ref. 31 and Ref. 32. Balachandran[12,33] developed a workflow management method capable of sequencing computational models with multiple inputs and outputs. Here, the workflow is automatically created given the input and output variables. However, the method requires a default workflow as input. The method proposed in this paper removes this limitation.

III. Proposed Approach

The architecture assessment, in particular, the system sizing and performance evaluation process can be seen as a sequence of five steps, as shown in Figure 3. Requirements and logical views are taken as input. Here, the logical view consists of logical flow and hierarchical descriptions of the system. In addition, computational (sizing) models of the system components must be available and are taken as input.

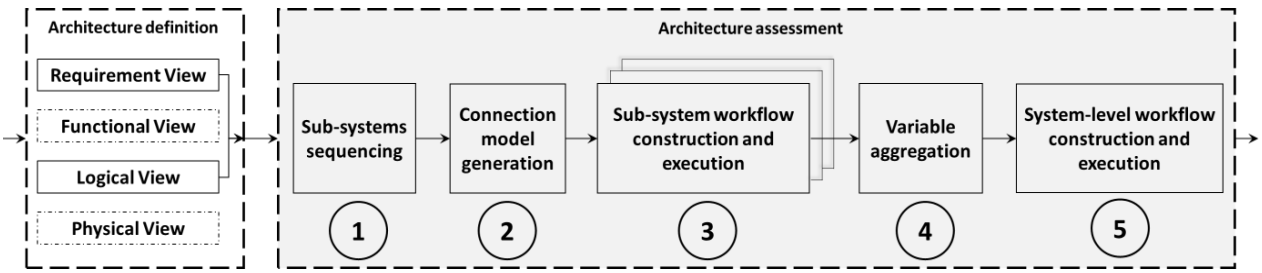


Figure 3 Steps involved in the assessment of the architecture.

In practice, simulation specialists will be given the task to evaluate system level performance, in line with the steps of Figure 3. As discussed in the introduction, new architectures can be created by modifying existing baselines. It is assumed here that the architectural changes involve addition/deletion of components or connections in the logical view of the architecture. A component in the logical view is assigned ports through which it is connected to other components. The port types are classified according to the physical entities "flowing" through the connections, for example, material, energy and signal[34]. The connections also define the direction of the physical flow among the components. It is assumed that connections can only exist between the same types of the ports.

Each port, depending on its type, has some parameters associated with it, for example, the electrical (energy) port is associated with voltage (effort) and current (flow) parameters. In Figure 4, the sets of parameters for the input and output ports are represented by $\mathcal{U}_j$ and $\mathcal{Y}_j$, respectively, as shown in Eq. (1) and (2). A component has its own parameters, referred to as component-parameters, represented by the set, $\mathcal{P}_j$ as shown in Eq. (3). For example, a compressor component may have three sets of parameters, i.e. parameters associated with, two input ports (one for air inlet and the other for required energy to compress air), an output port (for air outlet), and compressor itself. The set $\mathcal{U}$ is comprised of five parameters, associated to the two input ports, i.e. pressure (P), temperature (T) and mass flow rate (ṁ) of the air at the inlet of the compressor, and torque (τ) and rotational velocity (ω) at the input energy port. The set $\mathcal{Y}$ contains three parameters, associated with the output air-port, i.e. pressure (P), temperature (T) and mass flow rate (ṁ) of the air at the outlet. Finally, the set $\mathcal{P}$ may contain two parameters associated with the compressor components, i.e. compression ratio (γ) and weight (W).

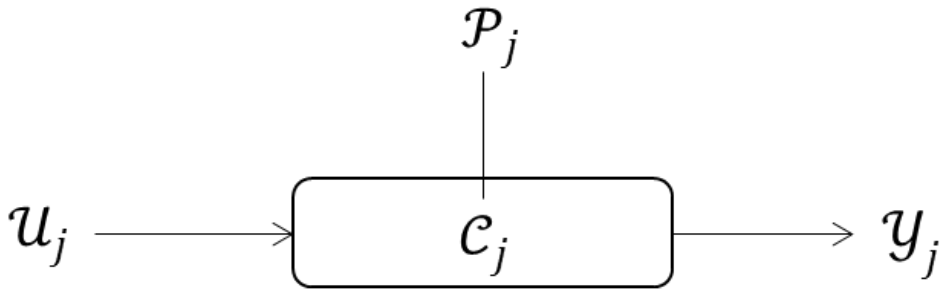


Figure 4. A logical view component parameters.

$$\mathcal{U}_j = \{u_{j_1}, u_{j_2}, u_{j_3}, \ldots\} \quad (1)$$

$$\mathcal{Y}_j = \{y_{j_1}, y_{j_2}, y_{j_3}, \ldots\} \quad (2)$$

$$\mathcal{P}_j = \{p_{j_1}, p_{j_2}, p_{j_3}, \ldots\} \quad (3)$$

Each function is fulfilled by a component as shown in Figure 5. Associated with each component are one or more computational models describing its behavior. The computational models use the parameters of the component, shown in Eq. (1), (2), and (3). These parameters are either inputs to or outputs from the models. In the computational view, all the parameters associated with the model are referred to as variables. Again, each variable is either an input to or an output from the model. It is also assumed that:

- Only single-function components exist in a logical view.
- The physical view of the architecture and relevant "transversal" computational models associated with it (e.g. thermal analysis) are not considered during the sizing process.
- The computational models related to each component contain the models associated with the function (i.e. the intended behavior of the component), and these models would be available.

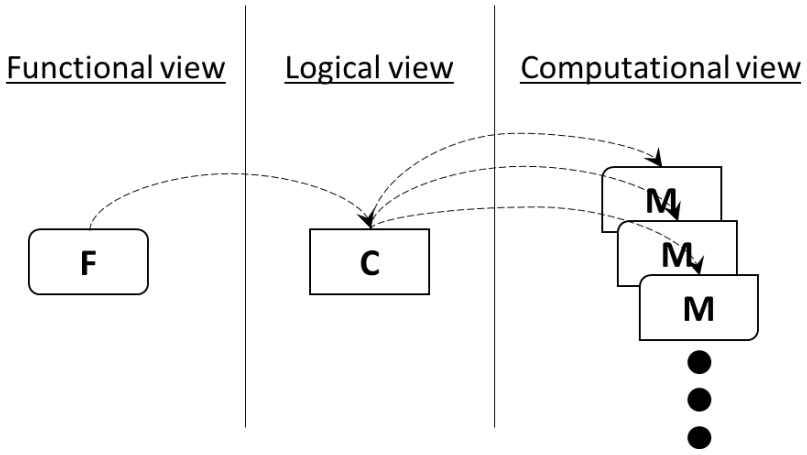


Figure 5. Mappings among different views.

The rest of the sections describe the enablers developed for each of the steps of the architecture assessment process as shown in Figure 3.

A. Step-1: Sequencing of sub-systems sizing tasks

The first step of the architecture assessment process is to sequence the sub-systems of a given logical view (step-1 in Figure 3). Figure 6 shows a generic example of a sub-system level representation of the system logical view.

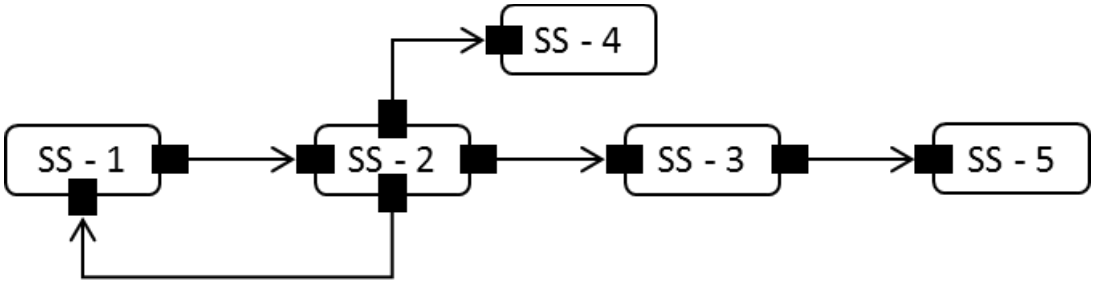


Figure 6. A sub-system (SS) level logical view of a system.

The method uses the source-sink approach as a basis for devising the sizing sequence. A source-sink relation between any two sub-systems is created by connecting relevant ports (Figure 7), for example, connecting the mass flow ports of the engine (bleed air) (sub)system and the de-icing (sub)system. Depending on the flow direction of the connection, one of the associated sub-systems is the source and other is the sink as shown in Figure 7. The rationale behind the sequencing of the sub-systems for their sizing is that the sink sub-system should be sized before the source sub-system. That is, the sink sub-system sets targets for the source sub-system during the sizing process.

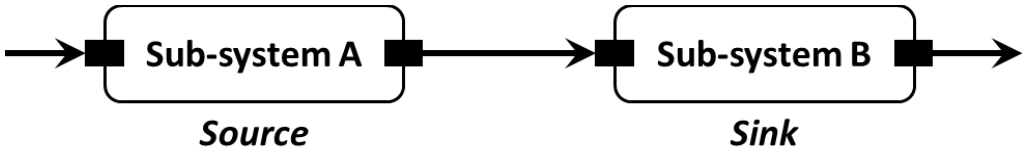


Figure 7. A source-sink relation between the sub-systems.

Following this approach, a source-sink Design Structure Matrix (DSM) of the logical view at sub-system level is created. The DSM is a square matrix where each row and column represents a sub-system. The matrix elements store the source-sink relations between the sub-systems. If a non-diagonal element is marked with '1', then the sub-system associated with the corresponding row is a source, whereas the sub-system associated with the column is a sink. The source-sink DSM of the logical architecture of Figure 6 is shown in Figure 8.

Given the source-sink DSM, the sizing sequence of the sub-systems is obtained by first finding the coupled sub-systems, also called Strongly Connected Components (SCCs) and creating a new matrix in which these SCCs are represented as single entries. This new matrix is then transformed into a Lower Triangular Matrix (LTM) form. Following that, the single SCC entries in the LTM are expanded back to their original state. The resulting matrix is referred to as the sequenced DSM. Then, the square sub-matrices formed at the diagonal of this sequenced matrix are utilized to extract the sequence. There are three types of the sub-matrices that can be formed at the diagonal of the matrix. These are shown in Table 1, where it is assumed that the sub-matrix is of dimension $p \times p$, $where\ p \leq n\ (DSM\ size)$. The first type is a diagonal matrix. The second type is a lower-triangular matrix, whereas the third type is a dense matrix, i.e., most (or all) of its non-diagonal elements are equal to non-zero.

Table 1 Types of sub-matrices formed along the sequenced DSM diagonal.

Sub-matrix Type	Mathematical description	
Type – 1	$\forall i \leq p\ \forall j \leq p \quad xij = 0$	i ≠ j
Type – 2	$[\forall i \leq p\ \forall j \leq i \quad xji = 0] \quad \wedge \quad [\exists i \leq p\ \exists j \leq i \quad xij \neq 0]$	i ≠ j
Type – 3	$\exists i \leq p\ \exists j \leq p \quad xij \neq 0$	i ≠ j

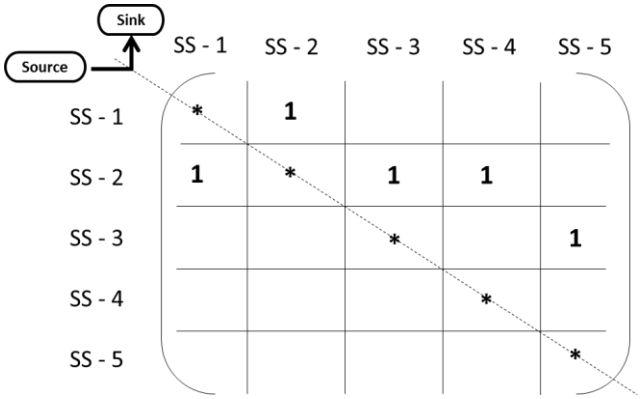


Figure 8. A source-sink DSM.

The sequence DSM obtained for the source-sink DSM of Figure 8 is shown in Figure 9. The figure also shows the three types of the sub-matrices that are formed at the diagonal. These sub-matrices are shown by solid, dashed and chain-lined rectangles, respectively.

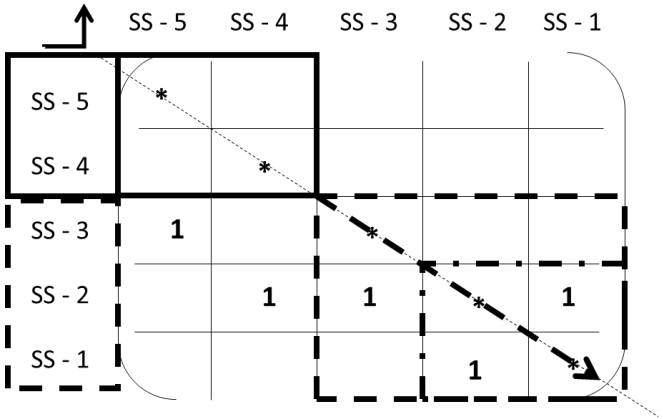


Figure 9. A sequenced DSM and associated sub-matrices at the diagonal.

The sizing sequence of the sub-systems is determined by the type of the sub-matrices along the sequenced DSM main diagonal. Table 2 gives the sub-system sizing sequence as per the sub-matrices of the sequenced matrix in Figure 9.

Table 2 Sub-systems sequence as per the sub-matrix types.

Sub-matrix type	Designation in Figure 9	Associated sub-systems (SS)	Sizing Sequence
Type-1	Solid-lined rectangle	SS – 5 and 4	Parallel
Type-2	Dash-lined rectangle	SS – 3 and, (SS – 2 and 1)	Sequential
Type-3	Chain-lined rectangle	SS – 2 and 1	Coupled (Iterative)

The execution sequence of the sub-system sizing tasks can be divided into stages for convenience. In particular, this staging can facilitate project and process management practices and enable appropriate allocation of computational resources, including parallel computation where possible.

The execution stages of the sizing tasks for the sub-systems in Figure 6, considering associated the sequenced DSM in Figure 9 are shown in Figure 10. The stages are established by tracing the sub-matrices along the sequenced DSM-diagonal from the top row to the bottom row. Here, SS-4 and SS-5 are associated with Type-1 sub-matrix; therefore, the sizing tasks associated with these sub-systems can be executed in parallel or in sequence. Sub-systems SS-1 and SS-2 are contained in the SCC and need to be executed iteratively. Furthermore, the parallel tasks (SS-4 and SS-5), SS-3, and the SCC (SS-1 and SS-2) are required to be executed sequentially because together they are associated with Type-2 sub-matrix accordingly they are grouped in stages.

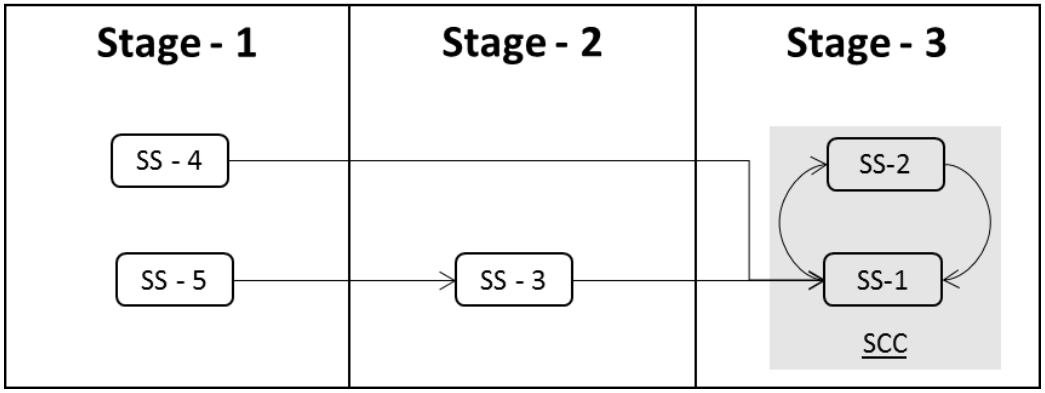


Figure 10. A sub-systems sizing sequence.

The sequencing method outlined above utilizes the following algorithms. The algorithm proposed by Xiao and Fei[35] is utilized to obtain a reduced DSM (with "lumped" Strongly Connected Components), to convert it into a Lower Triangular Matrix (LTM) and to obtain the sub-system sizing sequence. The algorithm identifies any Strongly Connected Components (SCCs) in the DSM and then reduces the DSM by aggregating the SCCs into single elements. The algorithm proposed by Tang et al.[36] is employed to obtain the SCCs of the original DSM.

Let's consider a system which contains m number of sub-systems (SS). Then the source-sink DSM is of dimension $m \times m$. Let 't' be the number of SCCs in the DSM. The SCC set is then:

$$\mathrm{S_{scc}} = \{\mathrm{SCC_1}, \mathrm{SCC_2}, \mathrm{SCC_3}, \ldots, \mathrm{SCC_t}\} \tag{4}$$

Each of the SCCs, (scc ϵ S_{scc}) is associated with two or more elements of the DSM - $DSM_{m\times m}$, and an element cannot be part of more than one SCC of the set S_{scc}. If the number of elements in the i[th] SCC (SCC_i) is given by *ElementOf*(SCC_i), then the total number of elements of $DSM_{m\times m}$ which are associated with SCCs, represented by z, is obtained:

$$z = \sum_{i}^{t} ElementsOf(SCC_i) \tag{5}$$

Considering each SCC as a single element, the original DSM is reduced to a matrix D^c of dimension $k \times k$, where $k = m - z + t$. To obtain the sizing sequence from the reduced DSM, an algorithm based on a theorem proposed by Xiao and Fei[35] is utilized. The algorithm obtains the execution stages, by identifying empty rows in the reduced DSM. To achieve this, an initial k-dimensional column vector containing all elements equal to 1 is created, and multiplied by the reduced DSM. The resulting product is a column vector. If the vector contains '1's, the (sub-systems at) rows corresponding to those indices can be executed in parallel, i.e. in a single stage. The sub-systems (associated with the empty row) identified in each of the iteration-steps correspond to one of the stages in the sequence. Once the elements associated with the rows are placed in some stage, then, these rows and their corresponding entries in the DSM are ignored in the next iteration. This is achieved by setting the next step-column vector containing zeros corresponding to the covered rows and '1's corresponding to uncovered rows (or non-zero elements). The process is repeated until there are no more non-zero elements. It should be noted that the number of stages is not fixed and is not dependent on the number of sub-systems. As explained above, the number of stages is determined by the connectivity of the sub-systems, which is obtained from the logical view of the systems architecture.

Following the derivation of the sizing sequence, the next step (see Figure 3) is to generate logical connections' models as described in next section.

B. Step-2: Generation of logical connections models

There are three types of logical connection arrangements possible, as shown in Figure 11. Type-I represents a one-to-one connection, where there is only one source-port and one sink-port. A source-port belongs to a source-component. Similarly, a sink-port belongs to a sink-component. A component is designated a source or a sink depending on the direction of the physical flow, i.e., by the arrow direction of the logical connection. Type-II connection denotes many source-ports, but only one sink-port, while type-III denotes the opposite, one source-port and many sink-ports.

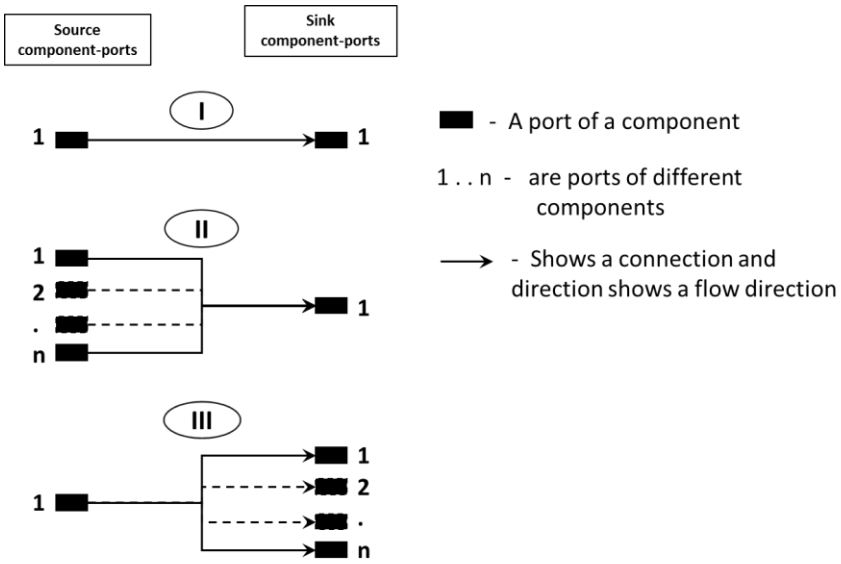


Figure 11. Types of logical connection arrangements.

There are three types of ports: material, energy and signal. Thus, in total, there are nine possible types of connections in the logical view. Depending on these combinations, connection models are developed which utilize the port-parameters. A simple example of three arrangement types for material (mass) flow ports, along with connection models for the three connection arrangements is shown in Figure 12. Here, the port-parameters, $\dot{m}$, P and T are mass flow rate, pressure and temperature, respectively. The connection models are created dynamically whenever architectural changes are made.

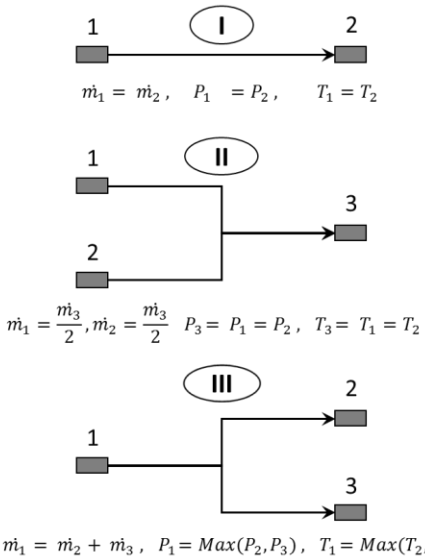


Figure 12. An example of connection models for a material flow.

C. Step-3: Construction of sub-system computational sizing workflow

Once the connection models are generated, the next step of the sizing process (see Figure 3) is to construct the computational sizing workflows employing these models and component computational models.

Here, the computational sizing workflow is modelled as a bipartite graph (BG). It is assumed that the computational (behavior) models associated with their corresponding sub-system components are available, whereas the logical connection models are generated dynamically as described above. During the sizing process, some of the variables of these models are assumed or designated as known and the rest are assumed unknown. The variables are designated as known when their values are available from the requirements placed on the system or target values set by the sink sub-system. To decide which unknown variable should be calculated from which model, the constraint management method[29] developed to solve a system of algebraic equations having single output is extended in this research to handle computational models with multiple-inputs and outputs.

The workflow (W) is represented by the directed bipartite graph (G):

$$\mathrm{W} = \mathrm{G}(\mathrm{M}, \mathrm{V}, \mathrm{E}) \tag{6}$$

where, M, V and E are sets of models, variables and edges, respectively. The following holds regarding W:

$$\mathrm{M} \cap \mathrm{V} = \emptyset, \tag{7}$$

where, $\emptyset$ is a null set. In addition, for an edge – e described by its end nodes – a and b,

$$\begin{gathered} \text{if } \mathrm{e} = (\mathrm{a}, \mathrm{b}) \ \wedge \ \mathrm{e} \in \mathrm{E} \text{ then,} \\ (\mathrm{a} \in \mathrm{M} \ \wedge \ \mathrm{b} \in \mathrm{V}) \ \oplus \ (\mathrm{b} \in \mathrm{M} \ \wedge \ \mathrm{a} \in \mathrm{V}) \end{gathered} \tag{8}$$

Here, $\oplus$ - is an 'exclusive or' operator.

Depending on available design information (i.e. known variable values), the variables set (V) is partitioned into two disjoint sub-sets as $\mathbb{U}$ and $\mathbb{K}$, called unknown and known variable sets, respectively, such that,

$$(\mathbb{U} \subset \mathrm{V} \ \wedge \ \mathbb{K} \subset \mathrm{V}) \ \wedge \ (\mathbb{U} \cup \mathbb{K} = \mathrm{V}) \ \wedge \ (\mathbb{U} \cap \mathbb{K} = \emptyset) \tag{9}$$

Once variable set (V) is partitioned into sets $\mathbb{U}$ and $\mathbb{K}$, next, undirected graph (UDG) is created deleting known variables nodes and their associated edges, that is,

$$\mathrm{UDG} = \mathrm{G}(\mathrm{M}, \mathbb{U}, \mathrm{UE}) \tag{10}$$

Here, $\mathrm{UE} \subset \mathrm{E}$ and the edges in the UDG are without direction.

To deal with the models with multiple outputs, duplicate model nodes are added in the UDG along with their corresponding edges. If a model has 'n' number of outputs then the duplicate model nodes added for this model will be 'n-1'. For example, consider a set of models associated with the set of components as shown in Figure 13. Figure 13(b) shows the default undirected bipartite graph for the computational models Figure 13(a).

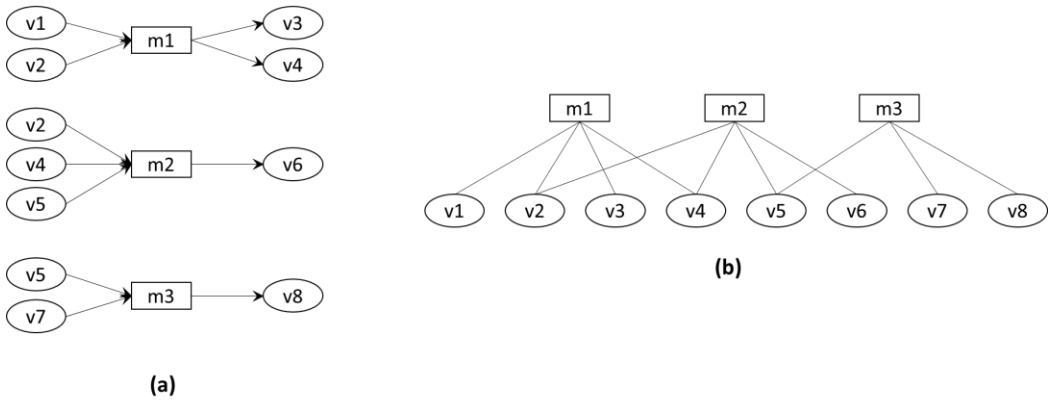


Figure 13. A set of models and their undirected BG representation.

As mentioned above, during the design process, some of the variables become known. Some of these are calculated at the sink sub-system and some are known from the requirements on the system. Thus assume that some of the variables are known. These are represented by green ellipses in Figure 14(a). The graph is rearranged, as shown in Figure 14(b), with known variable nodes above and unknown variable nodes below the model nodes. Thus the partition of the variable set into known and unknown variable set is:

$$\mathbb{K} = \{ v4, v6, v7, v8\} \tag{11}$$

$$\mathbb{U} = \{ v1, v2, v3, v5\} \tag{12}$$

Variable set V is the union of above two sets:

$$V = \mathbb{U} \cup \mathbb{K} \tag{13}$$

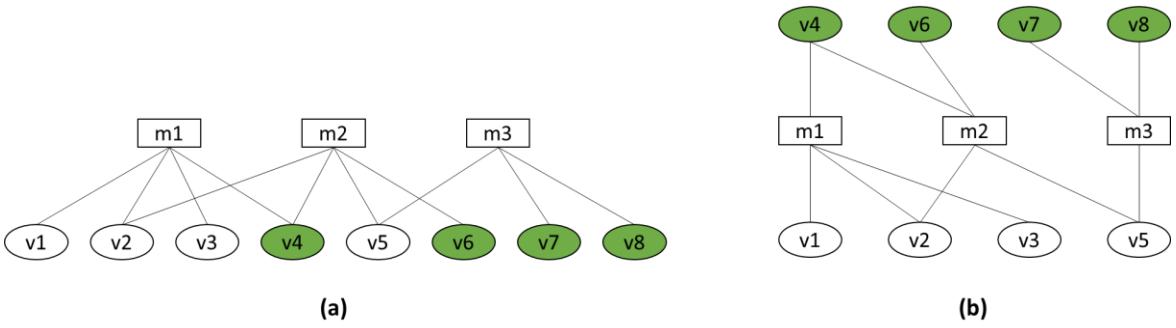


Figure 14. Known and unknown variables in a graph.

To deal with a model, m1 which has two output variables, duplicate model node (m1') is added, as shown in Figure 15.

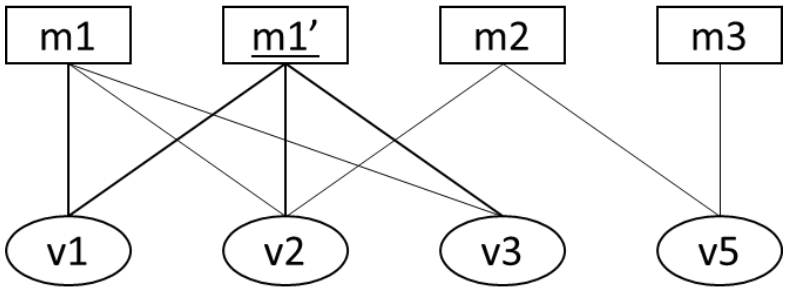


Figure 15. A UDG with duplicate nodes.

A maximum matching enumeration algorithm[37] is applied on the graph of Figure 15. The algorithm calls a recursive procedure which takes the graph as input. The procedure then divides the graph into $\boldsymbol{G}^{+}(\boldsymbol{e})$ and $\boldsymbol{G}^{-}(\boldsymbol{e})$ for the edge, 'e', and calls "itself" twice with each of these two new graphs as inputs, and so on. Here, the graph $\boldsymbol{G}^{+}(\boldsymbol{e})$ is created by deleting edge e and its nodes, and the edges associated with the deleted nodes, from the original graph (Figure 16 - lower left); whereas, $\boldsymbol{G}^{-}(\boldsymbol{e})$ is created by deleting only the edge from the original graph as shown in Figure 16 (lower right). The algorithm gives all the maximum matchings possible in the graph. Considering the graph of Figure 15, there are two possible maximum matchings, shown in Figure 17. In general, in the list of all the maximum matchings, there exist some which result in the same computational workflow. Here, both matchings result in identical workflow, shown in Figure 18. Such duplicates need to be filtered out from the matchings list produced by the enumeration algorithm. From the remaining matchings, the one with the minimum reversed models is selected for constructing the sizing workflow, and the latter is solved to find the unknown variables values. To solve the workflow, firstly, a model-model DSM is extracted from the workflow. This DSM shows the dependency of a model on the other models for its inputs. Next, SCCs contained in the DSM are found, and then, the SCCs and other models of the DSM are orchestrated. To solve the SCCs and any reversed models, mathematical (numerical) treatments such as fixed point iteration[12,14] and optimization[11,12] are employed.

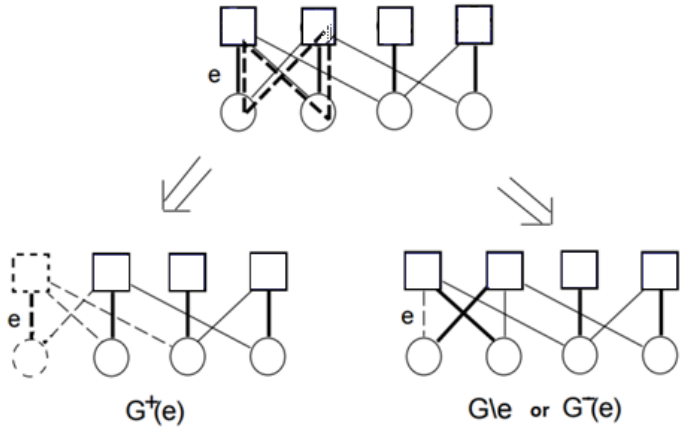


Figure 16. Definitions of $\boldsymbol{G}^{+}(\boldsymbol{e})$ and $\boldsymbol{G}^{-}(\boldsymbol{e})$ (adopted from the Uno[37])

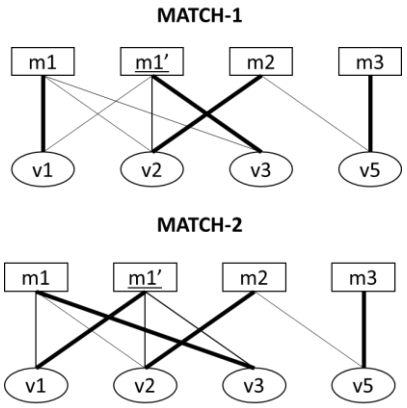


Figure 17. All possible maximum matchings.

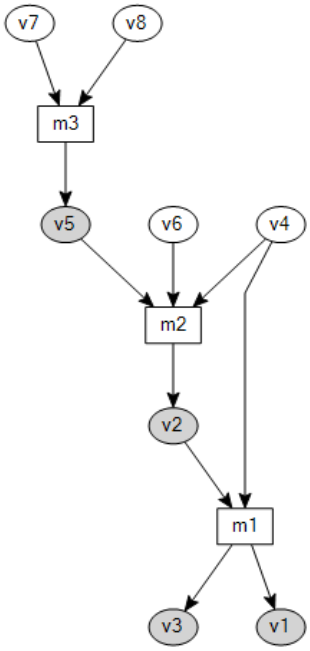


Figure 18. The workflow for selected matching.

D. Step-4: Aggregation of additive variables

Once all the sub-systems are sized, the next step is to aggregate variables at the system level (step-4 in Figure 3). Here, only additive variables are aggregated at the system level. For instance, to find the system weight, all the weights of the sub-system components are aggregated or summed up to find the system level weight. A recursive N-ary tree traversal is employed to traverse the pyramid of system hierarchy (Figure 1). In addition, the fundamental dimensions[38] are used to identify the relevant (additive) variables associated with the components of the (sub)system. For this purpose, an attribute called 'dimension' is included in the variable object. It is a 7-dimensional array storing powers of the fundamental dimensions as shown in Figure 19. The fundamental dimensions along with their unit in International System of Units (SI) are as follow: mass (kilogram (kg)), length (meter (m)), and time (second (s)), temperature (kelvin (K)), current (ampere (A)), luminous intensity (candela (cd)) and amount of substance (mole (mol)). For instance, a mass variable is represented with the fundamental dimension of mass, 'M' as '1' and the other dimensions are zero, i.e., [1, 0, 0, 0, 0, 0, 0]. Similarly, the power variable is represented as [1, 2, -3, 0, 0, 0, 0]. To aggregate particular variables during the N-ary traversal, the variables that are required to be aggregated are identified using the 'dimension' attribute of the variable.

All (additive) variables required at aircraft level are aggregated to find system-level performance. The top system level computational workflow is constructed using the same method, presented above, for the sub-system sizing.

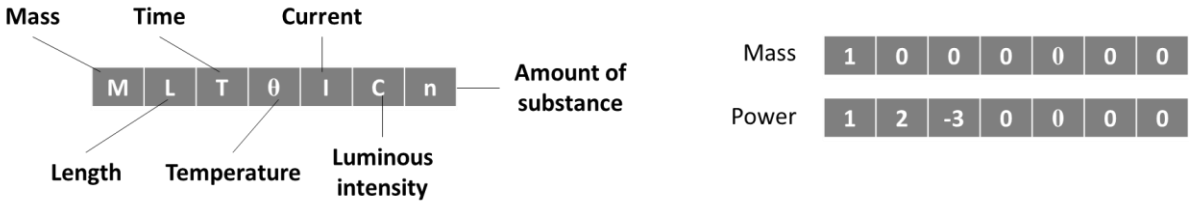


Figure 19 Fundamental dimensions concept. Left – the generic array of fundamental dimensions, Right – an example for Mass and Power.

Finally, in step-5 (see Figure 3), the system-level workflow is generated employing the same enabler developed for step-3 to find the system level performance.

IV. Evaluation

The academic literature on design research identifies three types of the evaluation. These include 'support', 'application' and 'success' evalution[39]. The support evaluation identifies whether the developed methods perform their intended tasks correctly. The application evaluation focuses on the ability of the method to address the targeted aspect of the research (here, conceptual design) as intended, while the success evaluation concentrates on the usefulness of the method, i.e. its ability to achieve the expected impact on the overall research-aim.

In this paper, evaluation is limited to the 'support evaluation'. For this purpose, the proposed method is compared with the existing methods through the employment of representative (usually simple) test-cases. As the proposed enablers overcome some of the limitations of the existing methods in the current research context, the comparison is performed for the situations where both the existing and proposed methods are applicable. For the other conditions, the results from the method are compared with the results obtained by manually performing the relevant tasks on a simplified test-case. The latter is simplified because it is difficult for a human to perform the tasks and to check the correctness of the results of a complex case.

The rest of the section is divided into three parts: evaluation of 1) the enabler for sequencing sub-systems for sizing, 2) the enabler for constructing computational sizing workflows of the sub-systems and 3) the enablers for generating the logical connection models and aggregating additive variables.

A. Sequencing of sub-systems sizing tasks

As mentioned earlier, the evaluation in this paper mainly focuses on verifying the correctness of the developed enablers. Therefore, they were investigated to find if the enabler gives the expected results for a given logical view. It should be emphasized at this point that, ideally the proposed method should be compared with a sizing sequence produced by an equivalent (competitor) method or with a result from an ongoing or a completed industrial program. To our best knowledge, there is no existing analog of the proposed method while a design sequence from an industrial program is not available at present. For this reason, the sequencing result is compared with a fixed aircraft power systems sizing the sequence proposed by Liscouët-Hanke et al.[6]

Recall that a system's logical view is taken as input in the first step of the sequencing process. To this purpose, a number of aircraft subsystems were considered in this test-case and are listed in Table 3. The corresponding logical view created in AirCADia Architect[2] (in-house architecting software tool) is shown in Figure 20. This view is given as input to the enabler sequencing the sub-systems. First, the enabler groups the elements of the logical view into aircraft sub-systems. To achieve this, a hierarchical representation (of logical view) of the aircraft components is employed. Once these sub-systems groups are identified, their inter-relations are established through the logical connections between elements (components) of these groups. In turn, the information of inter-relations between the groups is utilized to obtain the source-sink DSM of the sub-systems as shown in Figure 21. The DSM is given as input to the DSM sequencing algorithm to produce the sequence the sub-systems. GoDiagram[40], a diagramming library from Northwood is employed to represent the sequence in a graphical format. The sequence obtained is shown in Figure 22. The stages represent the sequential order (from left to right) in which the sub-systems should be sized.

From the sequence in Figure 10, it can be seen that the consumers, CAB, IPS, FPS etc., belong to the first stage, whereas the sources (ENG, APU, EPS etc.) fall in the last stage, while transformers (HPS, PPS etc.) are assigned to the intermediate stages. As per Liscouët-Hanke et al.[6], the sub-systems of the power systems group are sized in the following order: power consuming sub-systems, power distribution and transforming sub-systems, and finally power generation sub-systems. The sequence obtained using the proposed method reproduces the same sequence. In addition, the method can swiftly update the sequence given a change in the baseline architecture. Further evaluations of the method are planned including industry experts; these evaluations are not included in this paper.

Table 3 Aircraft sub-systems considered in the test-case.

No.	ATA Number	Sub-system Name
1	ATA 21	Environmental Control System (ECS)
2	ATA 23	Communications
3	ATA 24	Electrical Power System (EPS)
4	ATA 25	Cabin (CAB)
5	ATA 27	Flight Control System (FCS)
6	ATA 28	Fuel System
7	ATA 29	Hydraulic Power System (HPS)
8	ATA 30	Ice Protection System (IPS)
9	ATA 32	Landing Gear (LG)
10	ATA 33	Lightings
11	ATA 34	Navigation
12	ATA 36	Pneumatic Power System (PPS)
13	ATA 38	Water and Waste
14	ATA 49	Auxiliary Power Unit (APU)
15	ATA 72	Engine (ENG)

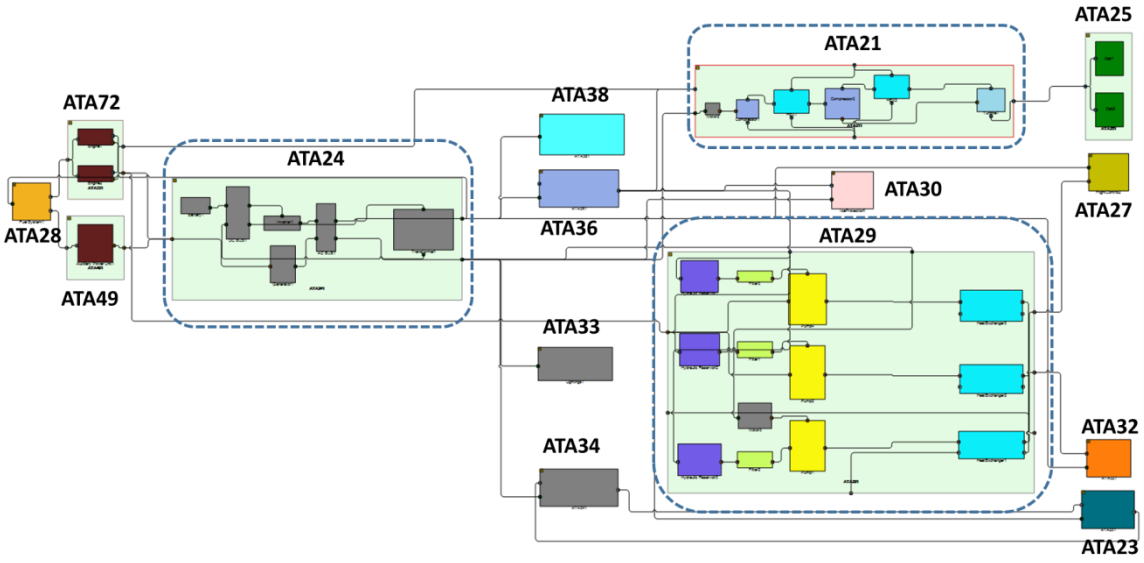


Figure 20. A logical view of the aircraft architecture.

			1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
			21	23	24	25	27	28	29	30	32	33	34	36	38	49	72
1	ECS	ATA 21	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
2	Communication	ATA 23	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0
3	EPS	ATA 24	1	1	1	0	1	1	1	1	1	1	1	1	1	0	0
4	CAB	ATA 25	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
5	FCS	ATA 27	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
6	Fuel System	ATA 28	0	0	0	0	0	1	0	0	0	0	0	0	0	1	1
7	HPS	ATA 29	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0
8	IPS	ATA 30	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
9	LG	ATA 32	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
10	Lightings	ATA 33	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
11	Navigation	ATA 34	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0
12	PPS	ATA 36	1	0	0	0	0	0	1	1	0	0	0	1	0	0	0
13	Water&Waste	ATA 38	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
14	APU	ATA 49	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0
15	ENG	ATA 72	0	0	1	0	0	0	1	0	0	0	0	0	0	0	1

Figure 21. Source-sink DSM of the logical view.

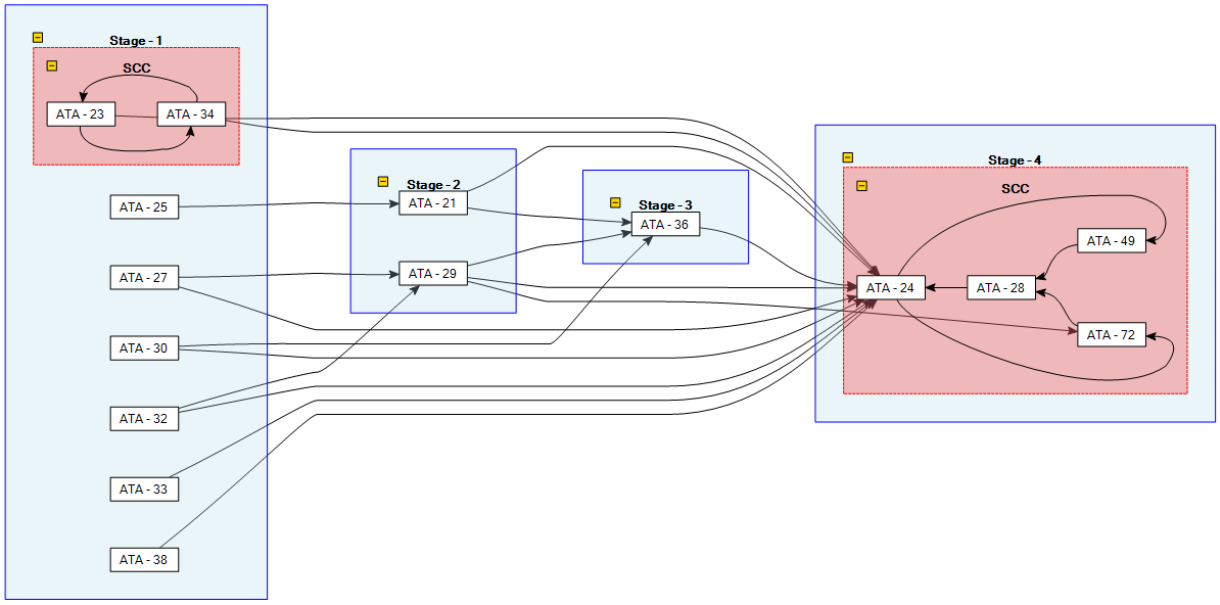


Figure 22. The sub-system sizing sequence for the given logical view.

B. Construction of the workflow

To verify the correctness of the proposed workflow construction method, it is compared with the existing methods such as the one devised recently by Balachandran[12]. It should be emphasized that Balachandran's algorithm requires a determined system of models to operate on. In the case of an under-determined system of models, a single computational workflow is created by randomly choosing the input variables from the set of (default workflow)

independent inputs. By contrast, the proposed here method generates all possible workflows for the under-determined system of models. In the proposed method, the independent inputs are referred to as the known variables.

Both methods are supplied with the same set of computational models, and results are compared. There are some cases, where Balachandran's method does not work because of its inability to deal with a system of models where variables can be outputs from more than one model. In those cases, the correctness of the workflows is manually checked. Then, the following features are compared:

- Computational workflow structure/topology results, obtained for the inputs in following three cases:
  - Determined (the number of independent variables is equal to the number of variables required for solving the models)
  - Under-determined (the number of independent variables is less than the number of variables required for solving the models)
  - Over-determined (the number of independent variables is greater than the number of variables required for solving the models)

The evaluation of this enabler is divided into two parts: 1) Comparison of the proposed method with Balachandran's method in the occasions where both methods are valid, and 2) Manual verification of the proposed method for the circumstances other than considered in the first part.

1. Comparison with Balachandran's method

In this part of the evaluation, a system of models considered is shown in Figure 23. The models' default inputs and outputs are shown in Table 4.

As per Balachandran, the determinacy a system of models is checked by the following conditions.

$$TNvar - NIvar - Noutmod = 0 \quad \text{(Determined)} \tag{14}$$

$$TNvar - NIvar - Noutmod > 0 \quad \text{(Under-determined)} \tag{15}$$

$$TNvar - NIvar - Noutmod < 0 \quad \text{(Over-determined)} \tag{16}$$

Where, $TNvar = Total\ number\ of\ variables$

$NIvar = Number\ of\ independent\ variables$

$Noutmod =$ Sum of number of outputs of the models

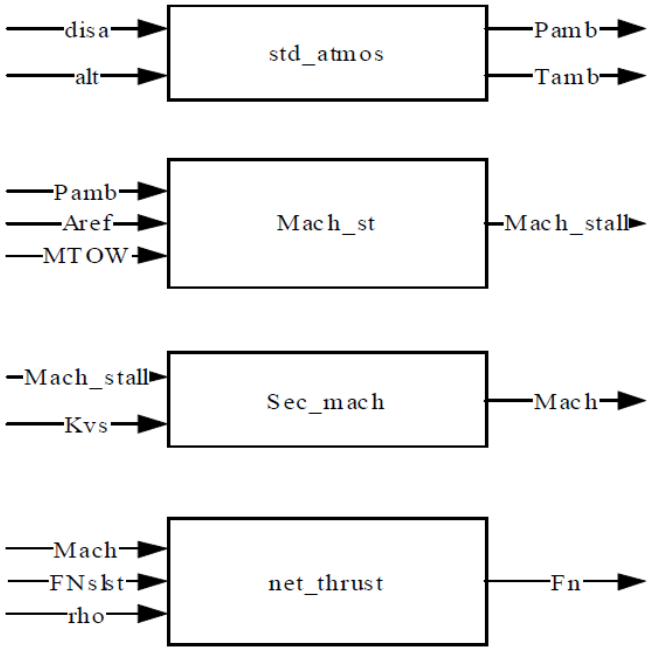


Figure 23 A system of computational models[12].

Table 4 Default workflow inputs and outputs.

Independent Inputs	Outputs
1. disa	1. Pamb
2. alt	2. Tamb
3. Aref	3. Mach
4. MTOW	4. Mach_stall
5. Kvs	5. Fn
6. FNslst	
7. rho	

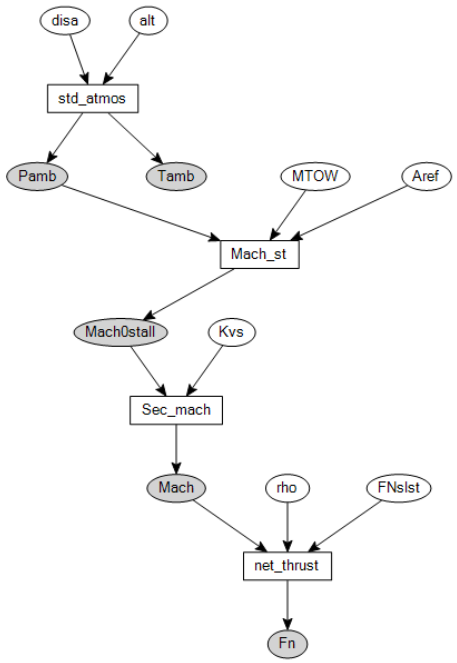


Figure 24 A default workflow.

Case-1: Determined system of models

The default workflow satisfies the condition in Eq. (14).
Here, $TNvar = 12$, $NIvar = 7$ and $Noutmod = 5$, and as per the Eq. (14),

$$12 - 7 - 5 = 0$$

Let's take a new set of independent inputs as shown in Table 5.

Table 5 Determined case: a new set of independent inputs.

Independent Inputs
1. Tamb
2. alt
3. Aref
4. MTOW
5. Kvs
6. FNslst
7. rho

Again the number of independent variables is 7. The condition in Eq. (14) is satisfied; therefore, the system of models is determined. The same set of models and independent inputs are given to both proposed method and Balachandran's method and the obtained workflows are compared. It is found that the workflows obtained by both

the methods are same. This confirms the correctness of the method. The new workflow produced by both the methods is shown in Figure 25. It should be noted that, in this workflow, the model *std_atmos* has a different set of inputs and outputs than its default condition shown in Figure 23. That is, the model is a reversed. Such models require a numerical optimization treatment to solve them, as described in the background section of this paper.

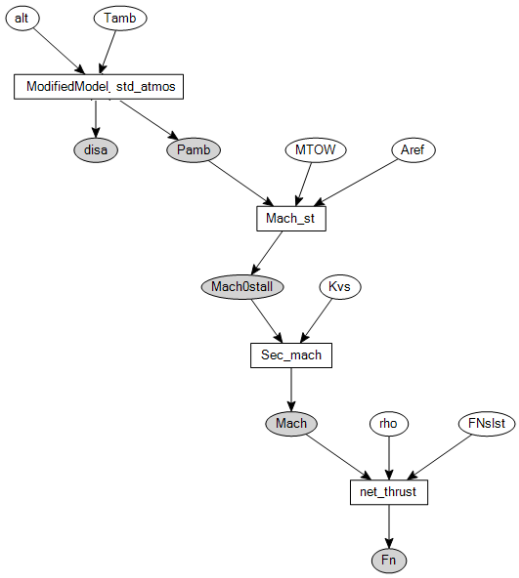


Figure 25 The workflow with a new set of independent inputs.

Discussion:

The Balachandran's method checks the determinacy of the system of models given a number of independent inputs and does not pay any attention to the independent variables themselves. That is, only the number of independent variables is not sufficient to check the determinacy of the system as there could be locally over-determined models for the given set of independent inputs. For instance, for the set containing 7 independent inputs – rho, disa, alt, Aref, MTOW, Kvs, and Pamb, the condition in Eq. (14) is still satisfied but the model – *std_atmos* is overdetermined. This model has two inputs and two outputs in the default condition as shown in Figure 23. However, with the above set of independent inputs, for this model, the independent inputs are three unlike in its default condition. Therefore, the expression in Eq. (14) is a necessary but not sufficient condition to check the determinacy of a system of models. The above issue is tackled by Datta[41] in his thesis; however, his method, an extension of the Balachandran's method, also requires a determined system of models at the start similar to the Balachandran's method.

In the presented method, the user is advised in the above occasions asking to relax some of the constraints on the locally over-determined model. That is, the list of relevant (concerning to the model) variables from the independent variables is displayed, and the user is asked to remove the variables (over-constraining the model) from the set of independent variables. This is discussed in detail while comparing the method in case of an overdetermined system of models i.e. Case-3.

Case-2: Under-determined system of models

The default workflow satisfies the expression in Eq. (15). The independent inputs are as given in Table 6.
Here, $TNvar = 12$, $NIvar = 6$ and $Noutmod = 5$, and as per the Eq. (15),

$$12 - 6 - 5 = 1 > 0$$

For an under-determined system of models, Balachandran's method produces single workflow given a determined workflow of these models. However, the proposed method generates all the possible workflows as shown in Figure 26, Figure 27 and Figure 28. The workflows also contain the workflow produced by Balachandran's method. In some of the workflows, there are model nodes (i.e. rectangles) with a text 'ModifiedModel'. These models need a numerical optimisation treatment to solve them as their inputs and outputs are different from their default condition. These are reversed models.

Table 6 Under-determined: a set of independent inputs.

Independent Inputs
1. disa
2. alt
3. Aref
4. MTOW
5. FNslst
6. rho

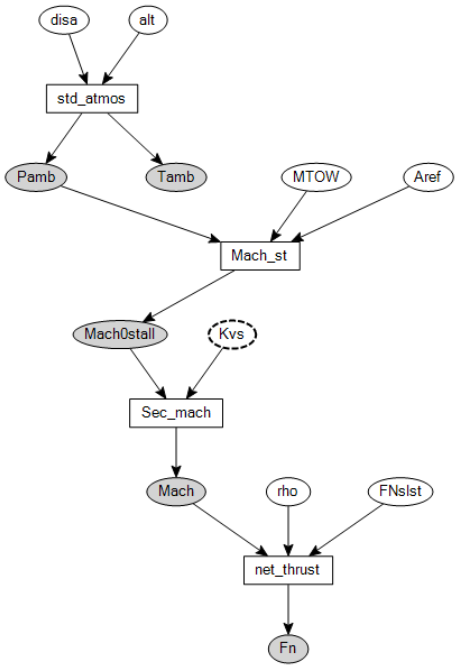


Figure 26 1st under-determined workflow.

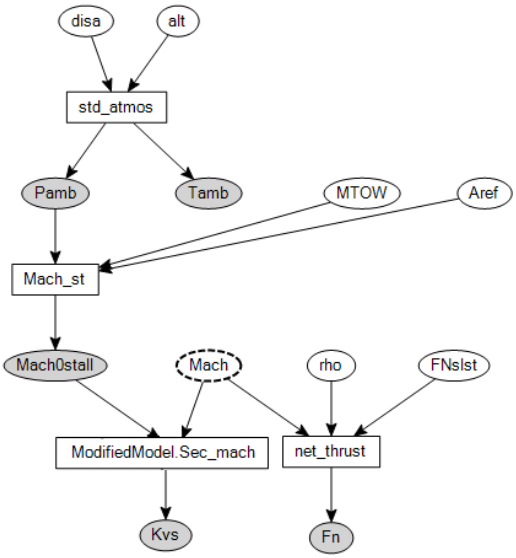


Figure 27 2nd under-determined workflow.

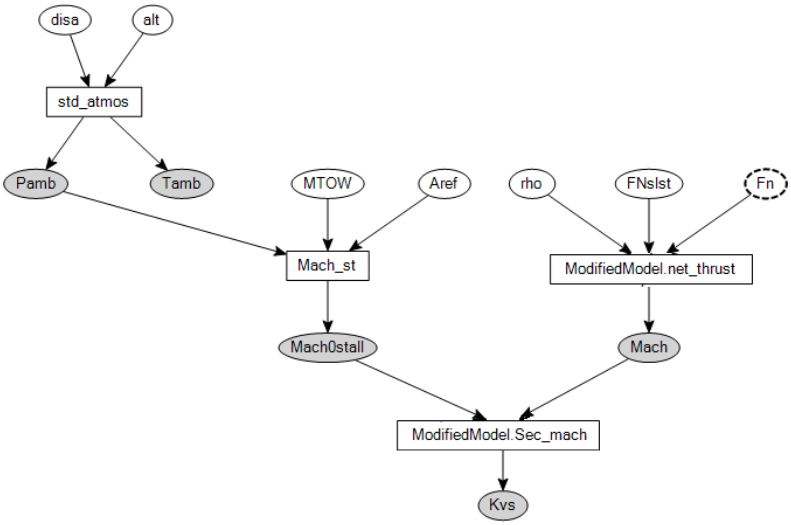


Figure 28 3rd under-determined workflow.

Discussion:

Although the proposed method produces all possible workflows, currently, in the presented method, there is not a way to choose a workflow out of the available workflows. Furthermore, the way method obtains the workflow, produces duplicate workflows. There is a need of an enabler that filters out the duplicate matchings listing only distinct workflows as well as orders the filtered workflows listing the workflows with minimum reversed models at the top. This would assist the user in choosing the workflow.

Case-3: Over-determined system of models

The default workflow satisfies the condition expressed in Eq. (16). The independent inputs are as given in Table 7.

Here $TNvar = 12,\ NIvar = 8$ and $Noutmod = 5$, and as per the Eq. (15),

$$12 - 8 - 5 = -1 < 0$$

In case of the over-determined system of models, Balachandran's method does not produce any workflow. In the same way, the proposed method also does not produce any workflow. However, the user is prompted with the list of over-determined models and asked to remove some of the independent inputs associated with the over-constrained models so that these models turn out to be determined.

Table 7 Over-determined: a set of independent inputs.

Independent Inputs
1. disa
2. alt
3. Aref
4. MTOW
5. Kvs
6. FNslst
7. Rho
8. Pamb

Discussion:

While dealing with a large number of models, it is difficult for a user to manually find the locally over-determined models of the system. In the proposed method, assistance is provided by listing the over-determined models and associated variables that would (potentially) need to be removed from the set of independent inputs in order to resolve the over-determined system.

2. Manual verification

This part of the evaluation is included because the method needs to be verified in the cases where existing methods are not applicable or available. For this purpose, the authors created computational workflows for a small set of computational models. The results obtained by the authors are then compared with the results obtained by the proposed computational method.

As mentioned above, it is possible that a variable could be an output from more than one model in the default workflow. In such a case, there is no direct comparison with existing methods[12]. Therefore the workflows are manually inspected for correctness. The models considered are shown in Figure 29. Here, it can be seen that variable, d is output from two different models, M1 and M2, whereas variable, e is output from models, M1 and M3.

Let us assume that variables a and b are known. The workflow obtained by the proposed method is shown in Figure 30. There is only one workflow possible for the given number of known variables. The authors manually obtained the same workflow. Now consider the under-determined case by reducing the number of known inputs (less than two). In this case, the known input is only one (i.e., variable, g). The method produced six possible workflows for this input. The authors could not derive all possible workflows but could verify the correctness of the (six) ones produced by the computer.

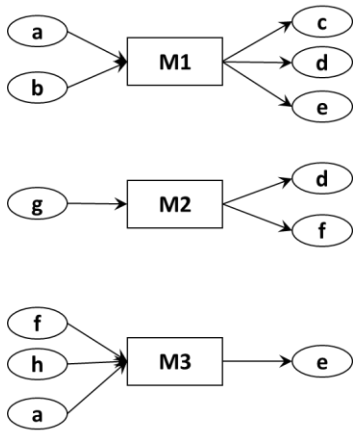


Figure 29 Computational models with one variable output of two models.

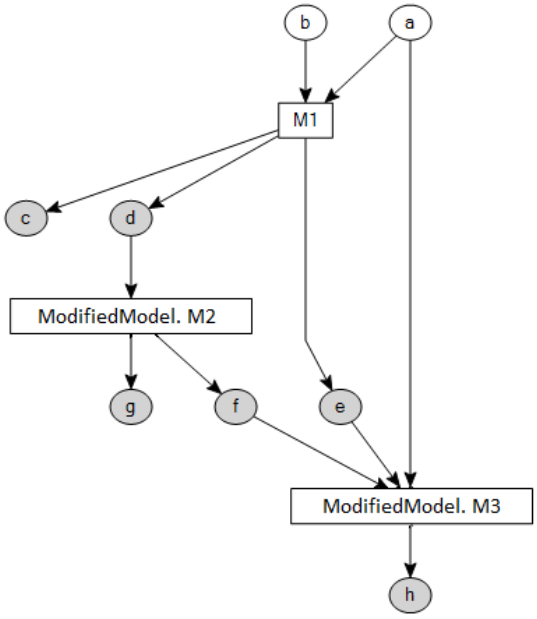


Figure 30 The workflow obtained with 'a' and 'b' as known variables.

Discussion:

Under the circumstances where the variables are output from more than one model, it is possible to find the number of independent inputs (i.e., known variables) by using Eq. (14) . For instance, the total number of variables ($TNvar$) in the system in Figure 29 is eight and total number of outputs from the models is six. Putting these numbers in Eq. (14) gives two independent inputs:

$$NIvar = TNvar - Noutmod = 8 - 6 = 2.$$

Event for this relatively simple test case the computer outperformed the human by finding more workflows. However, these had a different number of reversed (modified) models. As mentioned in the first part of the evaluation, ranking according to the number of reversed models will help the user to choose the workflow with minimum reversed models. This is important, as solving of reversed models needs a numerical optimization treatment, which results in increased computational cost.

C. Generation of logical connection models and aggregation of additive variables

The evaluation of the methods developed for generating connection models and aggregating additive variables are trivial and are not verified explicitly. The method for generating connection models is demonstrated through the logical view of a simplified Environmental Control System (ECS) architecture. The latter is created in AirCADia architect, as shown in Figure 31. Here, all the logical connection arrangements fall under the 'Type-I' (one-to-one) category. For each connection, depending on a number of associated port-parameters, there can be more than one computational model as shown in Table 8. In the table, the port-parameters, $\dot{m}$, P, T, and Pow are mass flow rate, pressure, temperature and rotational power, respectively.

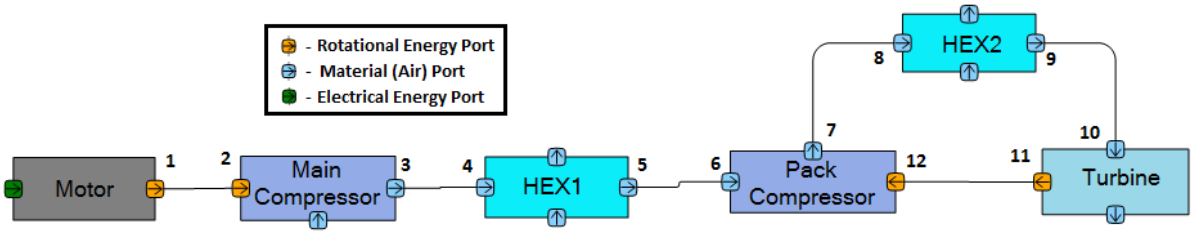


Figure 31 ECS logical view.

Table 8 Connection models.

Connections	Models
1-2	$Pow1 = Pow2$
3-4	$\dot{m}3 = \dot{m}4$ $P3 = P4$ $T3 = T4$
5-6	$\dot{m}5 = \dot{m}6$ $P5 = P6$ $T5 = T6$
7-8	$\dot{m}7 = \dot{m}8$ $P7 = P8$ $T7 = T8$
9-10	$\dot{m}9 = \dot{m}10$ $P9 = P10$ $T9 = T10$
11-12	$Pow11 = Pow12$

The implementation of the method for aggregating variables has been tested for correctness on a number of different systems hierarchies.

V. Conclusions and Future Work

Presented is an approach for automating the sub-system sizing tasks in (airframe) systems architecting, ultimately aiming to increase the efficiency and enable the interactivity of the process. The proposed framework is underpinned by and integrates a number of algorithms. The first one uses the system's logical architecture at sub-systems level as an input, to produce a source-sink representation. This, in turn, is used for the sizing sequence and for the dynamic creation of connection models. The latter combined with the component behavioral models comprise the computational sizing workflow. A bipartite graph maximum matching enumeration algorithm is employed to orchestrate the sizing workflow. Finally, a variable aggregation algorithm, which employs dimensional analysis (fundamental dimensions) and a recursive N-ary tree traversal are applied to compute the aggregated (additive) variable values at the system level.

The evaluation so far has been limited to ensuring the correctness of the developed algorithms. For that purpose representative test-cases were created, allowing both the computer and a human expert to produce the results for comparison. These results were compared also with the results from existing methods applied on the same test-cases. Where direct comparison was not possible, due to lack of (or deficiency of the) existing methods, the correctness was manually verified. The outcomes showed that the proposed enablers performed their intended tasks correctly.

Future work includes the implementation of the entire framework into a prototype software tool with a user-friendly graphical interface to enable evaluation by industrial experts. Further research will extend the proposed method to deal with multifunctional components. Also, as identified in the evaluation section, efficient algorithms for removing duplicate (identical) sizing workflows and for selecting the workflow with minimum reversed models will be developed. These algorithms will improve the overall efficiency and effectiveness of the method. Last, but not least, future research will include the consideration of different operational scenarios (e.g., entire mission or parts thereof) during the system sizing process.

References

[1]Kleiner, S., and Kramer, C., "Model Based Design with Systems Engineering Based on RFLP Using V6," S*mart Product Engineering: Proceedings of the 23rd CIRP Design Conference*, Bochum, Germany: Springer, 2013, pp. 93–102.

[2]Guenov, M., Molina-Cristóbal, A., Voloshin, V., Riaz, A., van Heerden, A. S., Sharma, S., Cuiller, C., and Giese, T., "Aircraft Systems Architecting-a Functional-Logical Domain Perspective," *16th AIAA Aviation Technology, Integration, and Operations Conference*, Washington, D.C.: 2016, pp. 3143.

[3]Chakraborty, I., Mavris, D. N., Emeneth, M., and Schneegans, A., "A System and Mission Level Analysis of Electrically Actuated Flight Control Surfaces Using Pacelab SysArc," *52nd Aerospace Sciences Meeting*, National Harbor, Maryland: 2014, pp. 381.

[4]Chakraborty, I., and Mavris, D. N., "Integrated Assessment of Aircraft and Novel Subsystem Architectures in Early Design," *Journal of Aircraft*, vol. 54, 2017, pp. 1268–1482.

[5]Chakraborty, I., Trawick, D. R., Mavris, D. N., Emeneth, M., and Schneegans, A., "A Requirements-driven Methodology for Integrating Subsystem Architecture Sizing and Analysis into the Conceptual Aircraft Design Phase," *14th AIAA Aviation Technology, Integration, and Operations Conference*, Atlanta, GA: 2014, pp. 16–20.

[6]Liscouët-Hanke, S., Mare, J. C., and Pufe, S., "Simulation framework for aircraft power system architecting," *Journal of Aircraft*, vol. 46, 2009, pp. 1375–1380.

[7]Liscouët-Hanke, S., Pufe, S., and Maré, J.-C., "A Simulation Framework for Aircraft Power Management," *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, vol. 222, 2008, pp. 749–756.

[8]Allison, J. T., Walsh, D., Kokkolaras, M., Papalambros, P., and Cartmell, M., "Analytical Target Cascading in Aircraft Design," *Aircraft Design*, 2005, pp. 1–9.

[9]Allison, J., Kokkolaras, M., Zawislak, M., and Papalambros, P. Y., "On the Use of Analytical Target Cascading and Collaborative Optimization for Complex System Design," *6th world congress on structural and multidisciplinary optimization, Rio de Janeiro, Brazil*, 2005, pp. 1–10.

[10]Kossiakoff, A., Sweet, W. N., Seymour, S. J., and Biemer, S. M., *Systems Engineering Principles and Practice*, John Wiley & Sons, 2011.

[11]Mathews, J. H., *Numerical Methods for Computer Science, Engineering, and Mathematics*, Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1986.

[12]Balachandran, L. K., "Computational Workflow Management for Conceptual Design of Complex Systems: An Air-vehicle Design Perspective," Ph.D. Dissertation, School of Engineering, Cranfield University, Cranfield, 2007.

[13]Balachandran, L. K., and Guenov, M. D., "Computational Workflow Management for Conceptual Design of Complex Systems," *Journal of Aircraft*, vol. 47, 2010, pp. 699–703.

[14]Chapra, S. C., and Canale, R. P., *Numerical Methods for Engineers: With Programming and Software Applications*, New York, NY, USA: McGraw-Hill, Inc., 1997.

[15]Ingram, C., Dendinger, T., Inclan, E., Charront, Y., Handschuh, K., Chakraborty, I., Garcia, E., and Mavris, D. N., "Integrating Subsystem Sizing into the More Electric Aircraft Conceptual Design Phase," *53rd AIAA Aerospace Sciences Meeting*, 2015, pp. 2015–1682.

[16]Kokkolaras, M., Louca, L., Delagrammatikas, G., Michelena, N., Filipi, Z., Papalambros, P., Stein, J., and Assanis, D., "Simulation-based Optimal Design of Heavy Trucks by Model-based Decomposition: an Extensive Analytical Target Cascading Case Study," *International Journal of Heavy Vehicle Systems*, vol. 11, 2004, pp. 403–433.

[17]Michalek, J. J., and Papalambros, P. Y., "Weights, Norms, and Notation in Analytical Target Cascading," *Journal of Mechanical Design*, vol. 127, 2005, pp. 499–501.

[18]Martins, J. R. R. A., and Lambe, A. B., "Multidisciplinary Design Optimization: a Survey of Architectures," *AIAA journal*, vol. 51, 2013, pp. 2049–2075.

[19]Paredis, C. J. J., Bernard, Y., Burkhart, R. M., Koning, H.-P., Friedenthal, S., Fritzson, P., Rouquette, N. F., and Schamai, W., "5.5. 1 An Overview of the SysML-Modelica Transformation Specification," *INCOSE International Symposium*, 2010, pp. 709–722.

[20]Tiller, M., *Introduction to Physical Modeling with Modelica*, Norwell, MA, USA: Kluwer Academic Publishers, 2001.

[21]Wang, R., and Dagli, C. H., "An executable system architecture approach to discrete events system modeling using SysML in conjunction with Colored Petri Net," *Systems Conference, 2008 2nd Annual IEEE*, 2008, pp. 1–8.

[22]Wang, R., and Dagli, C. H., "Executable System Architecting Using Systems Modeling Language in Conjunction with Colored Petri Nets in a Model-driven Systems Development Process," *Systems Engineering*, vol. 14, 2011, pp. 383–409.

[23]Jorrín, A., de Prada, C., and Cobas, P., "Ecosimpro and its el Object-oriented Modeling Language," *Proceedings of the 2nd International Workshop on Equation-Based Object-Oriented Languages and Tools*, 2008, pp. 95–104.

[24]Muñoz, L. J. Y., Vara, R. P., Bencomo, S. D., and Soria, M. B., "Comparison between Modelica 2.0 Y EcosimPro/EL 3.2," *2nd Meeting of EcosimPro Users, UNED, Madrid*, 2003.

[25]Pedersen, T. A., and Pedersen, E., "Bond Graph Modelling of Marine Power Systems," *Mathematical and Computer Modelling of Dynamical Systems*, vol. 18, 2012, pp. 153–173.

[26]Karniel, A., and Reich, Y., "From DSM-Based Planning to Design Process Simulation: A Review of Process Scheme Logic Verification Issues," *IEEE Transactions on Engineering Management*, vol. 56, Nov. 2009, pp. 636–649.

[27]Karniel, A., and Reich, Y., "Managing Dynamic New Product Development Processes," *17th Annual International Symposium of the International Council on Systems Engineering, INCOSE 2007 - Systems Engineering: Key to Intelligent Enterprises*, 2007, pp. 1053–1067.

[28]Chen, X., and Wu, H. Z., "Research on Conceptual Design of Virtual Prototyping for Complex Products," *Xitong Fangzhen Xuebao / Journal of System Simulation*, vol. 17, 2005.

[29]Serrano, D., "Constraint Management in Conceptual Design," Ph.D. Dissertation, Mechanical Engineering, Massachusetts Institute of Technology, 1987.

[30]Rudolph, S., and Bölling, M., "Constraint-based Conceptual Design and Automated Sensitivity Analysis for Airship Concept Studies," *Aerospace Science and Technology*, vol. 8, 2004, pp. 333–345.

[31]Friedman, G., *Constraint Theory: Multidimensional Mathematical Model Management*, Springer Science & Business Media, 2006.

[32]Buckley, M., Fertig, K., and Smith, D., "Design Sheet-An Environment for Facilitating Flexible Trade Studies during Conceptual Design," *Aerospace design conference*, Irvine, California: 1992, pp. 1191.

[33]Guenov, M. D., Tang, D., Lockett, H., and others, "Computational Design Process Modelling," *Proceedings of 25th International Council of the Aeronautical Sciences*, Hamburg, Germany: 2006.

[34]Stone, R. B., and Wood, K. L., "Development of a Functional Basis for Design," *Journal of Mechanical design*, vol. 122, 2000, pp. 359–370.

[35]Xiao, R., and Fei, Q., "Application of the Improved Method in Structural Modeling to Comprehensive Management of the Mine Bereaus," *Systems Engineering: Theory and Practive*, vol. 3, 1997, pp. 57–62 (in Chinese).

[36]Tang, D., Zheng, L., Li, Z., Li, D., and Zhang, S., "Re-engineering of the Design Process for Concurrent Engineering," *Computers & Industrial Engineering*, vol. 38, 2000, pp. 479–491.

[37]Uno, T., "Algorithms for Enumerating All Perfect, Maximum and Maximal Matchings in Bipartite Graphs," *Algorithms and Computation*, 1997, pp. 92–101.

[38]Sonin, A. A., *The Physical Basis of Dimensional Analysis*, Technical report, Massachusetts Institute of Technology: 2001.

[39]Blessing, L. T. M., and Chakrabarti, A., *DRM, a design research methodology*, Dordrecht; New York: Springer, 2009.

[40]"GoDiagram - Advanced Controls for .NET Diagrams," *Northwoods Software* URL: https://www.nwoods.com/products/godiagram/index.html [cited 30 November 2017].

[41]Datta, V., "Interactive Computational Model-Based Design: A Blackbox Perspective," Ph.D. Dissertation, School of Engineering, Cranfield University, Cranfield, 2014.



https://dspace.org/

Towards automating the sizing process in conceptual (airframe) systems architecting

Bile, Yogesh H.; Riaz, Atif; Guenov, Marin D.; Molina-Cristobal, Arturo

2018-01-13
AIAA
http://dx.doi.org/10.2514/6.2018-1067
Bile Y, Riaz A, Guenov MD, Molina-Cristobal A, Towards automating the sizing process in conceptual (airframe) systems architecting, Proceedings of 2018 AIAA/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference, AIAA SciTech Forum, 8-12 January 2018, Kissimmee, Florida, USA

Downloaded from a DSpace repository. https://dspace.org/