AIRCRAFT SYSTEMS ARCHITECTING: LOGICAL-COMPUTATIONAL DOMAINS INTERFACE

Marin Guenov*, Arturo Molina-Cristóbal*, Atif Riaz*,
Sanjiv Sharma**, Adrian Murton**, Judith Crockford**
* School of Aerospace, Transport, and Manufacturing, Cranfield University, MK43 0AL, UK
** Airbus Operations Ltd, Filton, Bristol, BS34 7PA, United Kingdom

Keywords: *Systems Engineering, System Architecture Definition and Assessment*

Abstract

Presented are interface specifications and implementations aimed at enabling interactive early stage systems architecting through tighter integration between architecture definition and assessment. Specifically, two approaches for interfacing the logical and computational domains are proposed. The first one employs graph theoretic principles to configure steady state computational workflows from information available in the logical domain. The second approach applies model transformation techniques for converting the logical flow views into Modelica models. An initial evaluation indicates that the first approach is faster and is better suited to early design studies while the second one is more comprehensive. Also outlined is the mapping of the fundamental elements of the logical and computational domains to the MoSSEC (Modelling and Simulation information in a Collaborative Systems Engineering Context) standard. A number of associations have been identified as potential modifications or further extensions to the MoSSEC object model. This will enable the proposed approaches to be implemented while employing MoSSEC as an underlying collaboration standard for model-based systems engineering.

1 Introduction

Anticipating future market demands and the surge in development of novel technologies, the aerospace industry has recognised the need for developing the means to rapidly and reliably integrate these technologies into the next generation viable product solutions. The work reported in this paper is a part of a related UK industry led research project, APROCONE [1], which aims to develop an advanced design environment to support the conceptual definition and evaluation of complex products, thus providing the foundation on which to achieve significant improvements within the high value design process.

As part of this process, architecture definition and assessment are the two distinct activities. According to Maier [2], architecture definition (conducted by architects) deals largely with unmeasurable entities using non-quantitative tools and methods such as decomposition of requirements and functions, allocation of solutions to functions, and specification of interfaces between solutions. By contrast, architecture assessment (conducted by simulation specialists) deals with tools and methods involving measurable quantities, such as orchestration and execution of computational models, component sizing and optimisation. Once the architecture is defined by the architect, it is passed to the simulation specialists, who analyse the performance of the architecture and pass key parameters back to the architect. In practice, these two activities are performed iteratively and are based predominantly on manual exchange of technical information. The problem with this manual approach is that if the architect modifies the architecture (based on results obtained from the simulation specialist), the whole process needs to be started again. Thus efficiency can be hindered significantly.

Within this context, the focus of the presented work has been on researching interface specifications aimed at enabling an interactive approach to early stage aircraft architecting. The objective is a tighter integration between architecture definition and assessment.

It is assumed that the systems architecting process takes place in four domains: Requirements, Functional, Logical and Physical – referred to as 'RFLP' [3]. Within the RFLP paradigm, architecture definition involves the requirement, functional, logical, and physical domains, while architecture assessment deals mainly with the logical, physical and computational domains, as shown in Fig. 1. Although the physical domain (especially information regarding the spatial layout) is required for architecture assessment, the scope of this work is limited to interfacing only the logical and computational domains, as shown in Fig. 1.

The rest of this paper is organised as follows. Background terminology is summarised in Section 2; State-of-the-art methods and tools, and associated research challenges are identified in Section 3. Two approaches for interfacing the logical and computational domains are described in Section 4. The interface specifications are evaluated in Section 5 with the help of a case study concerning the architecting of an environmental control system (ECS). Finally a summary and conclusions are presented and future work is outlined in Section 6.

2 Background

In this section a definition of some basic terms related to architecture definition and assessment is presented.

Logical Flow (View): A schematic diagram with components and their connections is referred to as logical flow view. The fundamental elements of the logical domain are components, ports, and connections.

Components: A component is a technical solution, e.g. part, subsystem, or even the whole system/product, which fulfils a particular required function.

Ports and Connections: A port describes the interface of a component with other components and the environment, including the type of the flow passing through the interface (e.g. energy, material, signal, etc.) and the direction, which can be either 'input' or 'output'. A connection describes the link between two or more components.

Parameter: A parameter is an engineering quantity that describes some characteristics of a component or port. For instance, compression ratio γ is an example of a parameter associated to the compressor component.

Computational Model: A computational model is a mathematical description of a component for predicting its behaviour or performance characteristics. Computational models can be static or dynamic: the former calculates the behaviour at steady-state

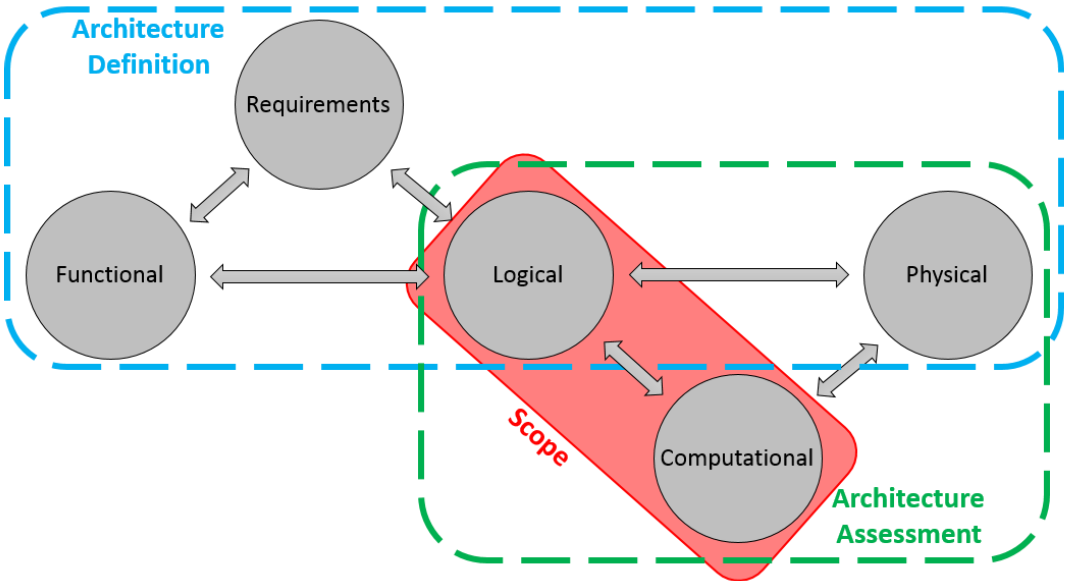

Fig. 1. Domains involved in architecture definition and assessment

(equilibrium) condition, whereas the latter accounts for the time-dependent changes in the component's behaviour.

Connection Model: The purpose of a connection model is to maintain consistency of the aggregable quantities 'transferred' between the ports of connected components (e.g. mass, pressure, temperature, etc.). Connection models are created dynamically, i.e. if the architect modifies links between components in the logical view, the corresponding connection models will be modified.

Computational Workflows: An ordered network of computational models intended to produce a meaningful result is generally referred to as computational workflow. Automatic (dynamic) methods [4] [5] can be employed to generate computational workflows.

Computational Domain: The collection of parameters, computational models, and workflow constitute a virtual domain, which is referred to as the computational domain in this paper.

3. Literature Review

A state-of-the-art review and challenges associated with interfacing architecture definition and assessment (logical to computational domain) are presented in this section.

3.1 Architecture Definition (Descriptive Modelling)

Various tools and languages have been proposed for descriptive modelling of systems architectures, such as, architecture analysis and design language (AADL) [6], systems modelling language (SysML) [7], logical flow view (LFV) [3] [8], object process methodology (OPM) [9]. Due to their relation to the presented below interface specification, only the Logical flow view and SysML are of concern to this work.

SysML is a general-purpose modelling language for specifying, analysing, designing, and verifying complex systems. It is a de-facto standard, proposed by OMG, for systems engineering and is widely used for architecture definition. SysML reuses a subset of UML2 [10], and defines its own extensions (new diagram types specific to systems modelling), with a total of nine diagrams. These diagrams are generally, categorized into four types, i.e. structure, behavior, requirements, and packaging. The structure diagrams, which describe the logical (schematic) view of the architecture, includes two diagrams, i.e. block definition and internal block.

3.2 Logical domain mapping to MoSSEC object model

The proposed MoSSEC standard (ISO/AWI 22071) [11] is designed to provide a capability to share Modelling and Simulation information in a collaborative Systems Engineering Context. This is to enable full traceability and re-use of modelling and simulation information throughout the product lifecycle and independent of the specific IT applications used across collaborating enterprises. The standard targets to cover a core subset of AP239 [12] systems engineering information content and related information services, that can be readily implemented and deployed to support engineering collaboration. The MoSSEC objects related to logical domain are listed in Table 1.

Table 1 : Mapping of logical domain elements to MoSSEC objects

Logical Domain Elements	MoSSEC Classes/Objects
Components	BreakdownElement
	BreakdownElementStructural Association
	BreakdownElementAssociati on
Ports	InterfacePortType
	InterfacePortInstance
Connections	InterfaceConnectionType
	InterfaceConnectionInstance

Three objects, i.e. BreakdownElement, BreakdownElementStructuralAssociation and BreakdownElementAssociation are primarily used to define the components of the logical domain, but are also used to define objects in the functional domain. The BreakdownElement objects represent the components, whereas the two association objects, BreakdownElementStructuralAssociation and BreakdownElementAssociation, represent the components relations. The BreakdownElementStructuralAssociation is used between elements in the same Breakdown to define the composition, e.g. if a component is comprised of two subcomponents, then there will be BreakdownElementStructuralAssociation objects between parent and child components. The BreakdownElementAssociation is used between elements in different breakdowns, e.g. if a component fulfils a particular function, then BreakdownElementAssociation is used to represent the mapping between function and component. The remaining four objects, i.e. InterfacePortType, InterfacePortInstance, InterfaceConnectionType and InterfaceConnectionInstance, are used to define components' connections (interfaces) to other components. The mappings between the logical domain elements and the MoSSEC objects are listed in Table 1.

It appears that in the current MoSSEC object model, a few of the components' associations, related to systems architecting, are missing, or need further usage guidance. For instance, suppose that the architect selects a component to fulfil a function, and that component has an associated additional required (derived) function. Using the BreakdownElementAssociation to describe this could lead to confusion. One proposal is to use the classifiedBy property available on all MoSSEC objects to classify the BreakdownElementAssocation as a "derived" association. This proposal needs further investigation to assess its suitability, and an agreed set of classifications for a collaboration context. Finally, MoSSEC objects "Component" and "ComponentAssemblyUsage" need to be investigated to see if they can be used to define a component type, e.g. where there are two engine components created which are of the same type (let's say, Turbofan). In this particular case, the same set of computational models can be used for all engines of the same type. Component and ComponentAssemblyUsage have not been exercised before so it is unclear how they should be used in to represent types.

3.3 Architecture Assessment (Analytical Modelling)

Architecture assessment involves generating mathematical (analytical) models of the architecture in order to estimate its performance. Several tools and languages are used to create mathematical models, e.g. Matlab/Simulink [13], Modelica [14], Mathematica [15], Amesim [16], etc.

Recently, there has been a strong interest in generating analytical (simulation) models automatically from the descriptive models [17][18] [19]. In most of these efforts, the UML profiling mechanism is used to embed simulation properties into SysML models. Simulation-specific profiles are employed in popular SysML modeling tools, such as MagicDraw [20] to annotate SysML models with simulation properties appropriate for the specific simulation environment. Next, enriched SysML models are transformed to executable simulation code for the specific environment. Model transformation languages, such as ATL [21] and QVT [22], are often utilized to transform SysML models to computational models represented in XMI, an XML representation language for UML/SysML models.

Although, the process of generating computational models is similar, it is not standardised. Also (full) automation is limited by the availability of pre-defined computational model libraries.

4 Interfacing Logical and Computational Domains

In this section two approaches for interfacing logical and computational domains are presented, as shown in Fig. 2. The approaches are not identical in the sense that the first is aimed at early design and utilises fast steady-state computational models, while the second one is aimed at employing dynamic models and is better suited for more detailed simulation. Both approaches assume that the requisite computational models (steady-state or dynamic) are available.

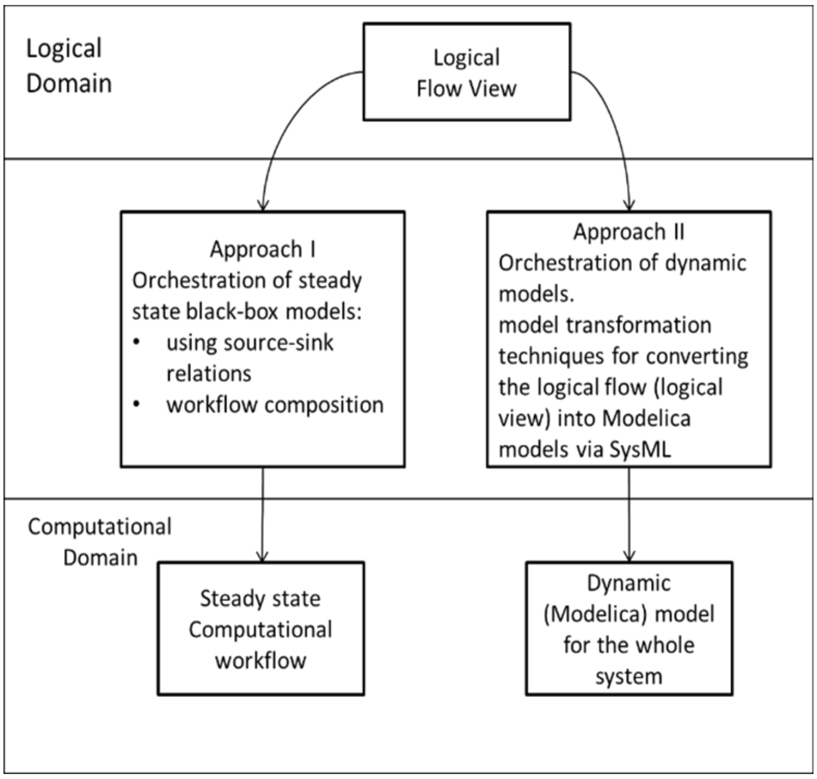
Logical Domain
Logical Flow View
Approach I
Orchestration of steady state black-box models:
• using source-sink relations
• workflow composition
Approach II
Orchestration of dynamic models.
model transformation techniques for converting the logical flow (logical view) into Modelica models via SysML
Computational Domain
Steady state Computational workflow
Dynamic (Modelica) model for the whole system

Fig. 2. Specification for interfacing logical and computational domains

4.1 Approach I

The input to this method is the logical flow view and the associated behaviour models for each component (which may be black boxes). The logical flow view is used to extract source-sink relations between the connected components and subsystems, which enable to determine the target values from one subsystem on other subsystems. The extracted source-sink relations are then used to obtain the workflow of the computational models. More details of the method are presented in reference [5].

This approach allows dynamic reconfiguration of the computational workflow, depending on the set of inputs/outputs specified by the systems architect. This feature is very useful for enabling design exploration.

4.2 Approach II

The second approach is comprised of two main steps. In the first step, the logical flow view is converted into SysML block definition [bdd] and internal block [ibd] diagrams. The second step is to employ SysML4Modelica profile to convert the SysML model to Modelica model.

For the second step, model transformation concept is used to automatically convert the SysML models into the executable models (e.g. Modelica [14] models). One of the common ways to transform SysML models into executable models is to use SysML profiles and extensions. SysML profiles and extensions transform the semantics of a simulation language/tool into the SysML semantics by defining stereotypes and tag definitions which are applied to specific model elements such as classes, attributes, operations and activities. Many such profiles have already been developed. The two most well-known profiles for linking SysML to Modelica are ModelicaML [23] and SysML4Modelica [18]. ModelicaML includes Modelica elements in UML but the problem with this profile is that the whole Modelica program has to be implemented in UML. By contrast, in SysML4Modelica, the Modelica code is automatically generated from a diagram in SysML and does not require to be directly written in SysML. This profile is employed in the second step of Approach II to transform SysML models into Modelica models.

5 Evaluation

The two approaches are evaluated via a common case study concerning the definition and assessment (sizing) of a baseline and alternative (electric) architecture of environmental control system (ECS) pack. As specified in Fig. 2 the process starts with the logical flow view depicted in Fig. 3. A conceptual sizing study involving the compressor, heat exchanger and turbine components is considered.

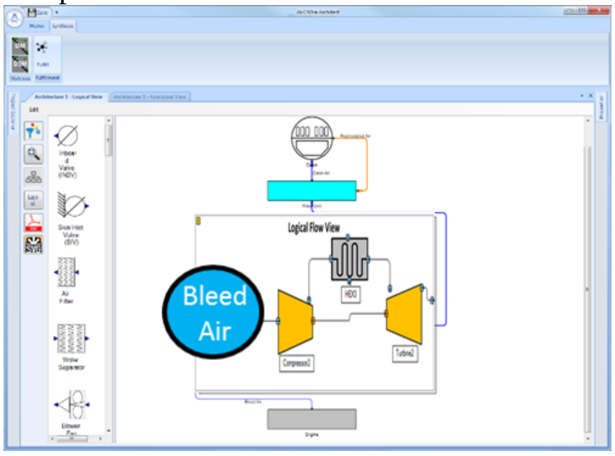

Fig. 3. Logical flow view of the conventional ECS or baseline architecture

A prototype tool, AirCADia Architect [8], is used to demonstrate Approach I. Approach II is implemented by employing AirCADia Architect (logical view), a SysML4Modelica plugin [18] for the software tool MagicDraw [20], and the Modelon environmental control library [24].

5.1 Approach I (Steady state model)

The component models of the ECS baseline are steady state and are based on the air-cycle bootstrap system. The relationships between temperatures of the air-cycle are given by the isentropic law applied to both compression and expansion [25]. The computational workflow is shown in Fig.4. The initial conditions and i/o nomenclature are shown in Table 2 and Table 3. In Table 2, the efficiency of the compressor, turbine and heat exchanger were set to be close to the ones used in the Modelica models of approach II. The flight conditions were assumed for take-off segment (sea-level altitude and Mach number of 0.3).

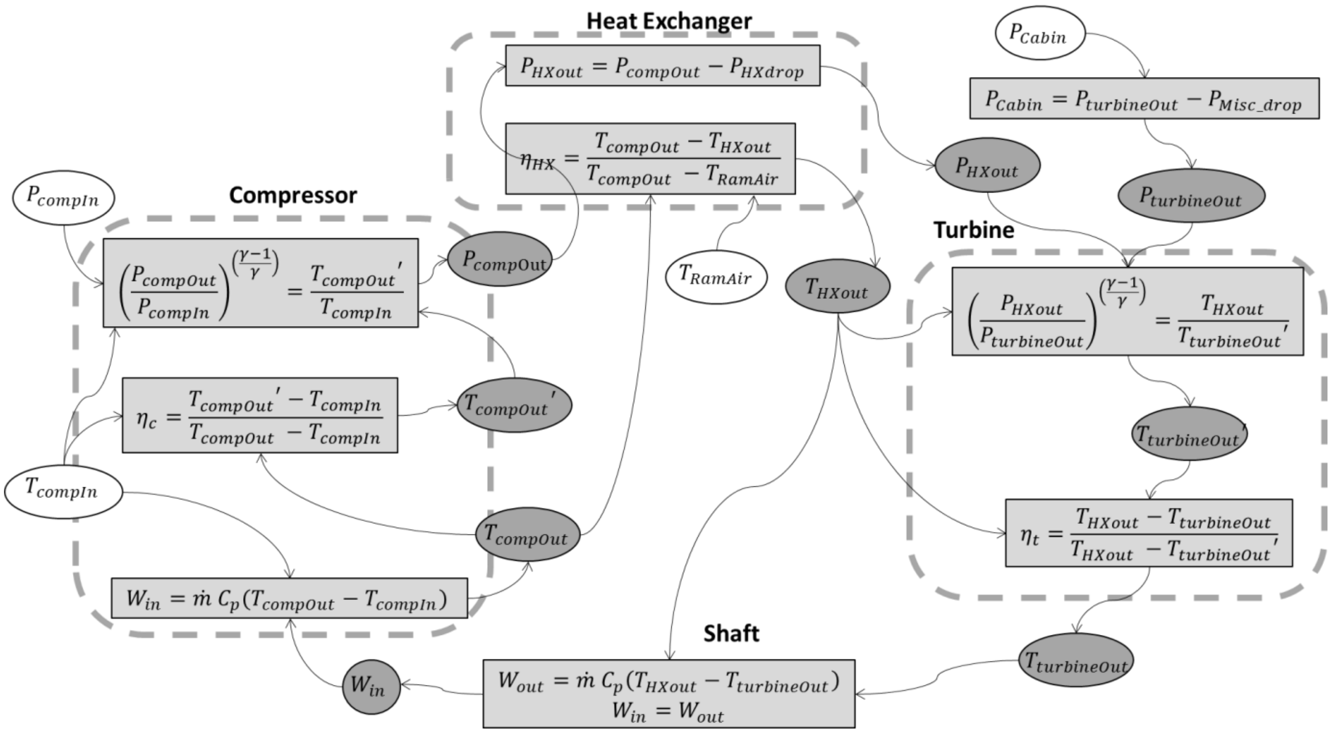
Heat Exchanger
$P_{HXout} = P_{compOut} - P_{HXdrop}$
$\eta_{HX} = \frac{T_{compOut} - T_{HXout}}{T_{compOut} - T_{RamAir}}$
P_{Cabin}
$P_{Cabin} = P_{turbineOut} - P_{Misc\_drop}$
P_{HXout}
$P_{turbineOut}$
P_{compIn}
Compressor
$\left(\frac{P_{compOut}}{P_{compIn}}\right)^{\left(\frac{\gamma-1}{\gamma}\right)} = \frac{T_{compOut}'}{T_{compIn}}$
$P_{compOut}$
T_{RamAir}
T_{HXout}
Turbine
$\left(\frac{P_{HXout}}{P_{turbineOut}}\right)^{\left(\frac{\gamma-1}{\gamma}\right)} = \frac{T_{HXout}}{T_{turbineOut}'}$
$\eta_c = \frac{T_{compOut}' - T_{compIn}}{T_{compOut} - T_{compIn}}$
$T_{compOut}'$
$T_{turbineOut}'$
T_{compIn}
$T_{compOut}$
$\eta_t = \frac{T_{HXout} - T_{turbineOut}}{T_{HXout} - T_{turbineOut}'}$
$W_{in} = \dot{m}\, C_p (T_{compOut} - T_{compIn})$
Shaft
$T_{turbineOut}$
W_{in}
$W_{out} = \dot{m}\, C_p (T_{HXout} - T_{turbineOut})$
$W_{in} = W_{out}$

Fig. 4. Workflow of the ECS steady state model

Table 2 : Initial conditions of the dynamic model and input variables to steady state model

Input variables	Numerical value
T_{RamAir} Ram air temperature [k],	296.75
T_{compIn} Bleed air temperature [K],	440.65
P_{compIn} Bleed air pressure [kPa],	512.5
$T_{compOut}$ Turbine output pressure [kPa]	166.151
η_c Compressor efficiency	0.64
η_{HX} Heat exchanger efficiency	0.86
η_t Turbine efficiency	0.825
γ Ration of specific heat of air	1.4
P_{HXdrop} Pressure drop [kPa]	1
P_{Misc_drop} Pressure drop due to friction [kPa]	36.151
P_{Cabin} Cabin pressure [kPa]	130

Table 3: Nomenclature of the steady state model output variables

Output variables	Symbol
Temperature after isentropic compression	$T_{compOut}{}'$
Compression temperature output	$T_{compOut}$
Compressor pressure output	$P_{compOut}$
Heat exchanger temperature output	T_{HXout}
Heat exchanger pressure output	P_{HXout}
Turbine pressure output	$P_{turbineOut}$
Temperature after isentropic expansion	$T_{turbineOut}{}'$
Turbine pressure output	$T_{turbineOut}$
Turbine or compressor work	W_{in}/W_{out}

The logical view (Fig5a) is input to the orchestration phase. Then workflow composition of the baseline ECS is created (Fig5b).

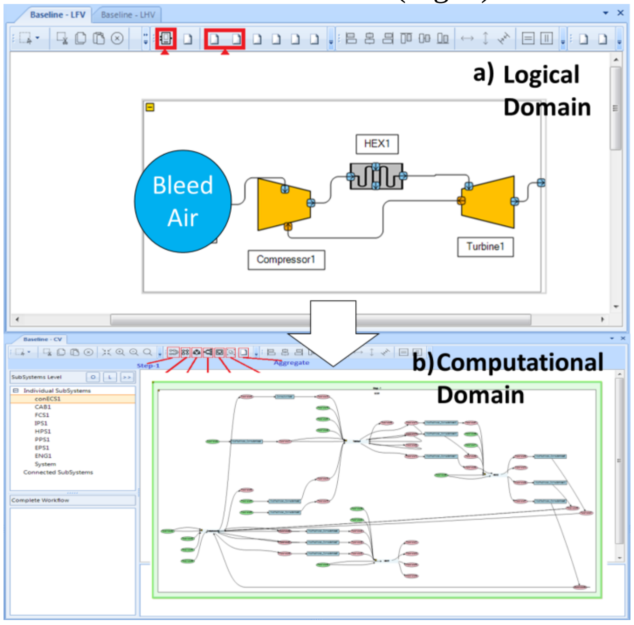

Fig. 5. Architectural views of the Approach I in AirCADia Architect

5.2 Approach II (Dynamic model)

Approach II starts with the logical view (Fig.6a). The first part (direction) of the interface is to convert from logical view to SysML diagrams. To this purpose, a parser utility program was implemented in AirCADia Architect. It converts the logical flow view from proprietary project file into an ".xml" format. This file contains all the data structures required for the SysML ibd diagram. Fig.6b shows a subsystem of the ECS architecture, which is the internal block diagram of the ECS-Pack.

The second step is to convert the ibd diagram (Fig. 6b) to Modelica. This was realised with the SysML4Modelica plugin, while the Modelica models for each component were employed from the Modelon library for Dymola (see Fig.6c). The assessment is performed by executing the Modelica models so that transient or time responses are obtained. The results are discussed in the next subsection.

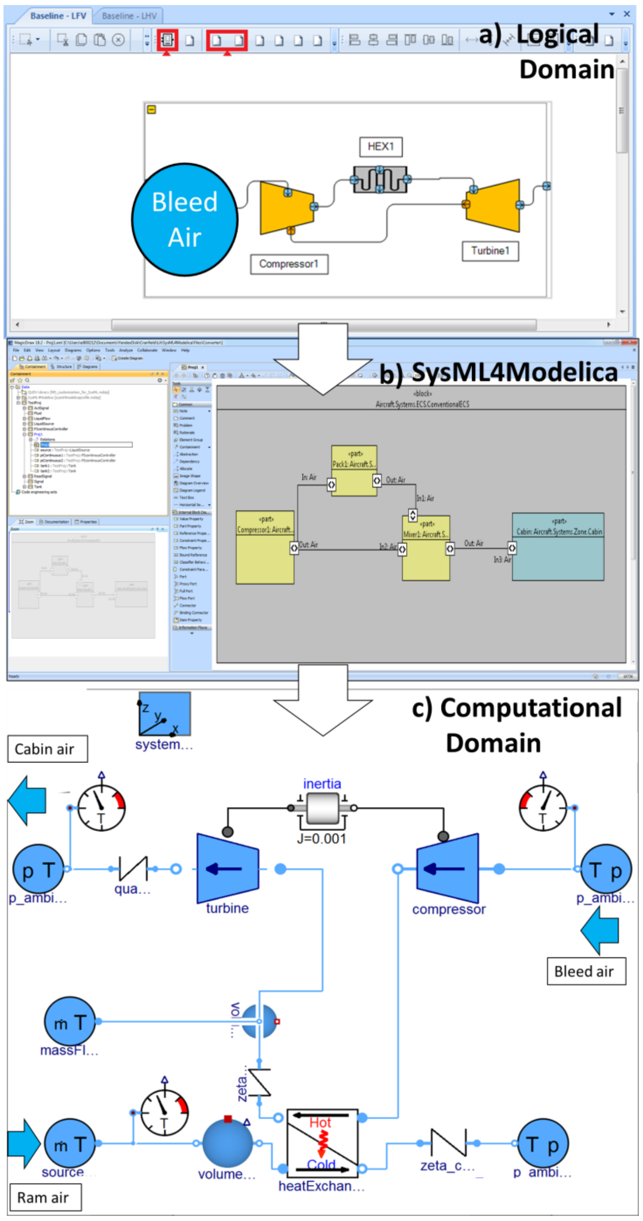

Fig. 6. Architectural views of the Approach II

5.3 Discussion

A semi quantitative comparison between the two approaches is presented in this section. It is based on two criteria described below.

5.3.1 Suitability for early stage systems sizing indicated by the computational effort

The pressure and temperature simulation results for each component of the baseline architecture are shown in Fig. 7, Fig. 8 and Table 4. The CPU-time (Intel i7-6820HQ, CPU @ 2.7GHz) for solving the computational workflow in approach I (steady state model) was 0.171s and 0.429s for approach II (dynamic model). That is, approach II is ~2.5 times more 'expensive'. For small-scale models this difference is negligible, however, it has been reported that for large-scale models with higher number of algebraic equations (1000 or more unknowns), Modelica tools face efficiency issues (high computational cost). This is due to the presence of non-linear algebraic loops and low performance of integrator (solvers) when a system has to track fast dynamics with small step sizes [26], [27].

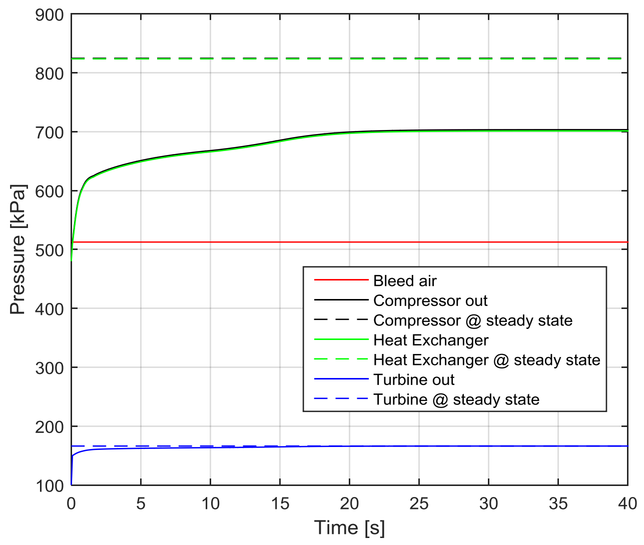

Fig.7. Pressure results of components from the baseline ECS architecture

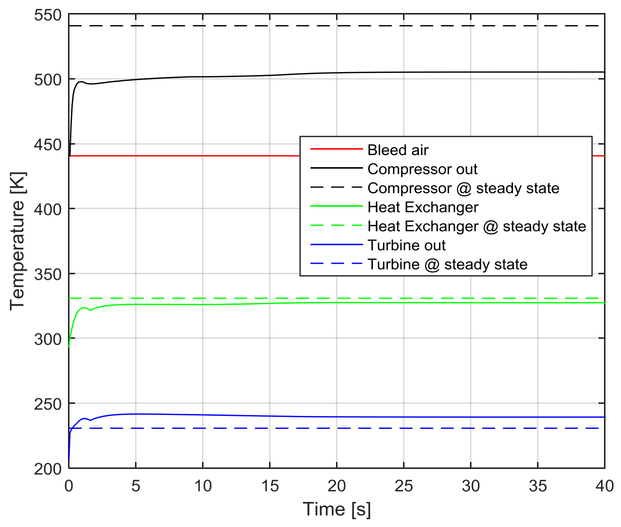

Fig. 8. Modelica model of the alternative (eECS) architecture

Table 4 : Numerical comparison of the pressures and temperature for each component of the baseline ECS architecture

	Pressure [kPa]			Temperature [K]		
	Dynamic model results at 40s	**Steady State model results**	**relative difference [%]**	**Dynamic model results at 40s**	**Steady State model results**	**relative difference [%]**
Compressor Output	703.33	824.6	~14.7%	505.218	540.9	~6.5%
Heat exchanger output	701.191	823.6	~14.8%	324.528	330.9	~1.9%
Turbine output	166.151	166.151	-	239.286	230.7	~3.5%

Except for the pressure output of the turbine, which is an initial condition to approach I, it can be seen, that the numerical results of the steady state models (approach I), and dynamic simulation (approach II) converge to different numerical values. The larger relative differences are with the pressure (~14.7%) and temperature (~6.5%) of the compressor output, the smallest relative difference is with heat exchanger (~1.9%) and the turbine (~3.5%) pressure. Since the Modelica compressor and turbine models are lookup tables of the efficiency maps, this numerical difference at steady state was expected. It has to be emphasised that finding an exact single-point solution is not the aim in early design. Rather, the designer's priority at this stage is to be able to swiftly explore possible alternative architectures [28]. The steady-state results here are sufficient to obtain an approximated size of the ECS-pack components. These considerations apply to other applications, for example, aircraft electrical networks [28].

5.3.2 Efficiency indicated by the amount of input information and effort required to orchestrate the sizing studies

It is known that replacing pneumatic systems with electrical one has potential benefits on power off-takes [29]. Thus, assume that during the conceptual phase there is a need to investigate an electrical ECS architecture. The changes required to be made to convert the baseline ECS into an electrical (e-ECS) one are performed in the logical view. Fig.9 depicts how the architect can interactively add or delete components by dragging and dropping.

After defining the alternative architecture, the computational workflow is created by using the automated link between architecture definition and assessment with approach I or II. In the case of approach I, the workflow composition was achieved within the AirCADia Architect design environment (Fig 9).

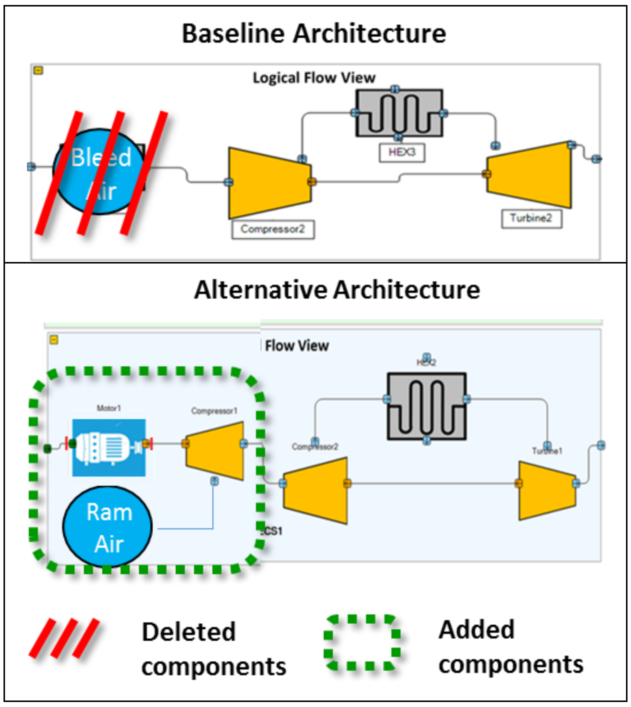

Fig. 9. Modifying ECS baseline architecture to an alternative architecture

Similarly, in approach II, the required changes are made in the logical domain. Then the above described two-step process of files conversion (as depicted in Fig.6) is applied. The resulting Modelica model of the eECS is illustrated in Fig.10.

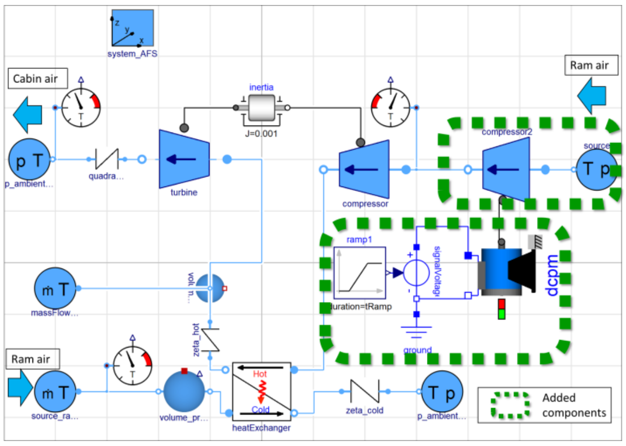

Fig. 10. Computational domain: Modelica model of the alternative (eECS) architecture.

A summary of the advantages and limitations of the two approaches is presented in Table 5.

6. Summary, Conclusions and Future Work

Presented in this document are interface specifications aimed at enabling interactive early stage aircraft architecting through tighter integration between architecture definition and assessment.

The focus is on specifications of the interface between the logical and the computational domains, where the latter is assumed to be an orchestration of analytical tools for (airframe) architecture assessment at systems and at aircraft level.

Two approaches are presented. The first one is intended for computations comprised of “black box” (fast/low order) computational models which allow the composition of dynamically reconfigurable computational workflows. The second approach employs model transformation techniques for converting the logical flow into Modelica models. It was demonstrated that the first approach is potentially faster, but the level of computation is limited to steady state models. This however is seen as appropriate for pre competitive or early conceptual studies. The second approach is computationally more expensive, but provides transient responses which are needed at preliminary design and system verification stages.

Future work includes the specification and prototype implementation of interfaces between the logical and physical and the physical and the computational domains, respectively. A more extensive investigation of scaling and associated computational cost of the two approached will be conducted.

Table 5: Summary of advantages and disavantages of the presented approaches

	Approach I	Approach II
Advantages	• Low computational, allowing fast architecture assessment • Less information intensive; can handle “black box” (fast/low order and steady state) computational models. • The effort required to orchestrate the sizing studies is low due to the ability to handle composition of dynamically reconfigurable computational workflows	• Higher Level of detail. Transient information is available and can be used for preliminary design stages. • Suitable for time response performance investigation and frequency domain studies. For example, at system (hardware) verification phase where the objective will be to demonstrate realistic integration of alternative architectures [28]. • Modelica is becoming a de facto standard. A large community is developing the language
Disadvantages	• Limited to steady state response. (However, for early conceptual stage this level of information is sufficient). • Sizing studies considering several operating conditions or entire mission are not yet available.	• Computational cost of dynamic simulations tends to be higher for large-scale models [26], [27]. • The orchestration of sizing studies depends on the availability of “white-box” (dynamic equations or source code) models.

Already underway is the mapping of the fundamental elements of the logical and computational domains to the MoSSEC standard. So far a number of areas have been identified for potential further investigation within the MoSSEC object model. This will enable the proposed specification to be implemented while employing MoSSEC as an underlying collaboration standard. The proposed areas for investigation include a) derived function, b) components to parameters mapping, c) components to computational models mapping, and d) component types.

7 Acknowledgments

The research leading to these results has received funding from the Aerospace Technology Institute (ATI) in the UK, under the Advanced Product Concept Analysis Environment (APROCONE) project (Ref no. 113092).

References

[1] "Advanced Product Concept Analysis Environment (APROCONE) - Aerospace Technology Institute (ATI) Research Project."

[2] M. W. Maier, "Developments in System Architecting," in *Proceedings of ICECCS '96: 2nd IEEE International Conference on Engineering of Complex Computer Systems (held jointly with 6th CSESAW and 4th IEEE RTAW)*, 1996, pp. 139–142.

[3] S. Kleiner and C. Kramer, "Model Based Design with Systems Engineering Based on RFLP Using V6," in *Smart Product Engineering, Proceedings of the 23rd CIRP Design Conference, Bochum, Germany, March 11th - 13th, 2013*, 2013, pp. 93–102.

[4] L. K. Balachandran and M. D. Guenov, "Computational workflow management for conceptual design of complex systems," *J. Aircr.*, vol. 47, no. 2, pp. 699–703, 2010.

[5] Y. Bile, A. Riaz, M. D. Guenov, and A. Molina-Cristobal, "Towards Automating the Sizing Process in Conceptual (Airframe) Systems Architecting," in *AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials*, 2018.

[6] P. H. Feiler, D. P. Gluch, and J. J. Hudak, "The architecture analysis & design language (AADL): An introduction," 2006.

[7] OMG, "Systems modeling language (sysml) specification—version 1.5," 2017.

[8] M. Guenov *et al.*, "Aircraft Systems Architecting - a Functional-Logical Domain Perspective," in *16th AIAA Aviation Technology, Integration, and Operations Conference*, 2016.

[9] D. Dori, *Object-process methodology: A holistic systems paradigm*. Springer Science & Business Media, 2011.

[10] OMG, "Unified Modeling Language (UML) 2.0 Specification," 2005.

[11] "Modeling and Simulation information in a collaborative Systems Engineering Context 'MoSSEC.'" [Online]. Available: http://www.mossec.org/.

[12] "ISO 10303-239 'Product Life Cycle Support.'" [Online]. Available: http://www.ap239.org/. [Accessed: 16-May-2018].

[13] "MathWorks Matlab/Simulink." [Online]. Available: https://www.mathworks.com. [Accessed: 16-May-2018].

[14] "Modelica - Object-oriented modeling language."

[15] "Wolfram Mathematica." [Online]. Available: https://www.wolfram.com/mathematica/. [Accessed: 16-May-2018].

[16] "Simcenter Amesim." [Online]. Available: https://www.plm.automation.siemens.com/global/en/products/simcenter/simcenter-amesim.html. [Accessed: 16-May-2018].

[17] O. Batarseh and L. F. McGinnis, "System modeling in SysML and system analysis in Arena," in *Proceedings - Winter Simulation Conference*, 2012.

[18] "SysML-Modelica Transformation SysML4Modelica Profile."

[19] G. D. Kapos, V. Dalakas, M. Nikolaidou, and D. Anagnostopoulos, "An integrated framework for automated simulation of SysML models using DEVS," *Simulation*, vol. 90, no. 6, pp. 717–744, 2014.

[20] "MagicDraw." [Online]. Available: https://www.nomagic.com/products/magicdraw. [Accessed: 16-May-2018].

[21] "ATL - a model transformation technology." [Online]. Available: https://www.eclipse.org/atl/. [Accessed: 16-May-2018].

[22] "OMG MOF Query/View/Transformation QVT Specification Version 1.1." [Online]. Available: https://www.omg.org/spec/QVT/1.1. [Accessed: 16-May-2018].

[23] "ModelicaML UML Profile."

[24] "Dassault Systèmes 3DExperience Release 2018x." [Online]. Available: https://www.3ds.com/products-services/3dexperience/. [Accessed: 23-Apr-2018].

[25] R. C. Arora, *Refrigeration and air conditioning*. PHI Learning Pvt. Ltd., 2012.

[26] F. Casella, "Simulation of large-scale models in modelica: State of the art and future

perspectives," in *Proceedings of the 11th International Modelica Conference September 21-23*, 2015, pp. 459–468.

[27] F. Jorissen, L. Helsen, and M. Wetter, "Simulation speed analysis and improvements of Modelica models for building energy simulation," in *Proceedings of the 11th International Modelica Conference*, 2015, pp. 59–69.

[28] M. R. Kuhn and Y. Ji, "Modelica for large scale aircraft electrical network VandV," in *Proceedings of the 10 th International Modelica Conference; March 10-12; 2014; Lund; Sweden*, 2014, no. 096, pp. 747–756.

[29] T. Jomier, "Final public More Open Electrical Technologies (MOET) Technical Report," 2009.

Copyright Statement