ICAS 2020 SHANG HAI
International Council of the Aeronautical Sciences
32nd Congress of the International Council of the Aeronautical Sciences
September 6-10, 2021
Pudong Shangri-La, Shanghai, China

CLOUD-BASED ENVIRONMENT FOR AIRCRAFT DESIGN COLLABORATION

Xin Chen, Atif Riaz, Marin D. Guenov

Cranfield University, Cranfield, MK43 0AL, United Kingdom

Abstract

Presented in this paper is an outline specification for a cloud-based collaborative aircraft design environment. It reflects particular industrial needs, such as capabilities required for remote workflow orchestration and design coordination. The architecture of the prototype application is presented along with initial implementation details of deploying commercial-off-the-shelf (COTS) cloud solutions. The usefulness of this prototype is evaluated with an aircraft sizing use-case and an airframe-engine design use-case. The former is utilized to test the cloud-based workflow orchestration capability while the latter is conducted to demonstrate remote design collaboration. The initial results are promising and indicate that the prototype application (employing COTS technologies and cloud providers) can be used to enable collaborative optimization studies between distributed design teams.

Keywords: Cloud Computing, Aircraft Design, Distributed Design, Workflow Orchestration, Design Collaboration

1. Introduction

Aircraft design is a highly complex system engineering process, which involves the integration of major components and numerous sub-systems. In practice, the overall aircraft design problem can be decomposed hierarchically into sub-problems, conducted by different designers and domain experts. These designers/experts may belong to multiple departments or even separate organizations, which underlines the heterogeneous and distributed nature of the process and the associated computing resources.

While traditional design collaboration relies heavily on workshops and review meetings, the rapid development of Information Technology (IT), has enabled extensive research on distributed (collaborative) design during the last three decades, with many collaborative design tools developed, especially in recent years. However, there remain a few challenges, related in particular to the utilization of the newly emerged cloud computing technologies: 1) There is still a lack of an integrated cloud-based tool for flexible orchestration of workflows and studies. 2) Current collaborative tools have been focused predominantly on the connections between models, while less attention has been paid to the interactions between designers. 3) Uncertainty regarding the risks and opportunities of deploying commercial-off-the-shelf cloud solutions in industry. These challenges have motivated the ongoing research for a cloud-based collaborative aircraft design environment as part of the COLIBRI project [1].

In this paper, we will mainly address the first challenge, which can be further divided into the three objectives: The first objective is to propose an approach/architecture to share computational models and general engineering capabilities (e.g. workflow orchestration, optimization, visualization capabilities) remotely for geographically distributed companies/design teams. The second objective is to identify the corresponding technology stacks (e.g. web frameworks, development toolkits, and communication protocols) to implement the cloud-based design environment. The last objective is to test the feasibility of employing commercial-off-the-shelf cloud solutions. The outcome is a cloud-based prototype application named “AirCADia Nebos”. This prototype will also be used as a platform to facilitate further research for tackling the other two challenges.

The remaining part of the paper is structured as follows: Section 2 reviews the state-of-the-art of

cloud computing technologies, along with existing research on distributed and collaborative design. Section 3 presents the specification for the proposed cloud-based design environment. In Section 4, the prototype tool is demonstrated with an aircraft and engine design use-case. Finally, summary, conclusions, and future work are outlined in Section 5.

2. State-of-the-Art

This section presents the background and relevant research on cloud-based design environment. In Section 2.1, we will briefly review the basics concepts of cloud computing, along with some technologies corresponding to IT implementation. Section 2.2 reviews existing research on cloud-based design methodology, tools, and applications.

2.1 Terminology Definition

2.1.1 Cloud Computing

There is no single universally accepted definition of cloud computing. However, there is a consensus is that it delivers computing as a service on demand via the internet [2]. One of the rigorous definitions, adopted in this research is given by the U.S. National Institute of Standards and Technology (NIST) [3]:

Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.

Cloud computing releases the burden on an enterprise or organization to operate its own IT system, therefore reduces the total cost and enables the enterprise to focus more on its original business. For example, instead of purchasing and maintaining its own physical servers, an enterprise can employ remote virtual servers from cloud providers. In case of computing demand changes, the virtual servers can be rescaled dynamically, and the enterprise only needs to pay for what it has used. The more professional and centralized management from the service provider will also reduce the risk of server failure and data loss, as the virtual servers are normally distributed and redundant.

2.1.2 Service, Service Oriented Architecture (SOA), Microservices, and Web Services:

According to the World Wide Web Consortium (W3C), a service is defined as “abstract resource that represents a capability of performing tasks that represents a coherent functionality from the point of view of provider entities and requester entities” [4]. Within the context of cloud computing, there are three typical service models depending on the type of services provided [3], namely: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS).

Based on the philosophy of considering capabilities as services, two related concepts are: Service Oriented Architecture (SOA) and Microservices Architecture. Both concepts are architecture design styles which employ loosely coupled functional components to achieve an overall target. The difference is mainly in the scope of deployment [5].

SOA is an enterprise-wise concept [6], and can be defined as “a paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains” [7]. The services in SOA can be multiple independent applications or even more general entities which are beyond IT implementations (e.g., a business activity [8]).

On the other hand, the microservices architecture is an application-wise concept, and can be regarded as an interpretation of SOA in the development of a single software. A microservice refers exclusively to a computer process which is designed to perform a single specific function and can be invoked separately. Within one application, microservices can be written in different languages and are loosely connected through Application Programming Interfaces (API). This distinguishes SOA from the monolithic architecture, where the components communicate via internal data structures and cannot be executed independently [9].

Finally, a web service is “a software system designed to support interoperable machine-to-machine interaction over a network” [4]. From this perspective, web service can be considered as a specific form of API using the web technology. One or more microservices can be deployed as a web service to expose their functionalities remotely.

We use the term “microservice architecture” for single software implementation, while SOA is used in the context of a collaborative design project, because the latter involves multiple partners, design activities, and design (software) tools.

2.1.3 Application Programming Interface (API)

As mentioned earlier, microservices are loosely connected through Application Programming Interface (API). Currently there are three major API options (for web services): Remote Procedure Call (RPC), Simple Object Access Protocol (SOAP), and Representational State Transfer (REST). All the three options can be operated with the HTTP protocol [10].

The RPC is a relatively old technology based on the client-server model. It was developed to extend local (program to program) procedure callings to remote addresses [11]. Compared with the other two options, it requires a tighter coupling between subsystems, therefore it has now been gradually replaced with the SOAP technology. SOAP is a protocol for transferring XML based messages. In general, a SOAP message consists of three parts: “an envelope that defines a framework for describing what is in a message and how to process it, a set of encoding rules for expressing instances of application-defined datatypes, and a convention for representing remote procedure calls and responses” [12]. REST is currently the most popular solution for building microservices-based applications. Different from the former options, it is an API design style rather than a fixed standard. The central philosophy of REST is to employ HTTP commands (GET, POST, CREATE, READ, UPDATE, and DELETE) to handle resources, which are exposed with Uniform Resource Locators (URL) [13]. Compared with the other two options, it is light-weight, efficient, and flexible.

2.1.4 Data communication

Commonly used options for data communication between microservices include plain text files, comma-separated values (CSV), Extensible Markup Language (XML) [14], and Javascript Object Notation (JSON) [15].

The plain text file is easy to create since it does not require any predefined structures. However, this may cause error and inconsistency during data exchange. In addition, it is less interpretable for a computer to extract useful information. For instance, to get the value of a variable, either the location is specified beforehand (which leads to low flexibility) or the file needs to be searched line by line with key words matching (which leads to low efficiency). The CSV format can be regarded as a text file with a very simple structure for tabular data (columns delimited using commas as indicated by its name). However, it cannot represent complex data structures such as functions, objects, and hierarchies, etc. To manipulate those data structures, the XML and JSON are widely used and very similar in their capabilities. Both these two formats have consistent syntaxes and hierarchical structures for data representation. While being effectively machine-interpretable, they are also convenient for humans to understand. Compared with JSON, XML has a few additional functions such as supporting namespace, accommodating more data types (e.g. images), etc. However, JSON is easy to generate and to process due to its simplicity. In addition, JSON is readily interpretable by JavaScript function, while XML needs to be parsed, which makes it less efficient in data transmission. As a result, although XML has a longer history, JSON is now used more often and is gradually replacing XML in practice.

2.1.5 Database Management Systems (DBMS) and Object Relational Mappers (ORM)

Database is the commonly used to store information over the cloud. A database is defined as “a shared collection of logically related data and its description, designed to meet the information needs of an organization”, while a Database Management Systems (DBMS) is defined as "a software system that enables users to define, create, maintain, and control access to the database." [16].

In general, a database and its corresponding DBMS can be classified as relational or non-relational. In a relational database, the data are structured as relations, where “a relation is physically represented as a table with columns and rows” [16]. The rigorous mathematical foundation was originally proposed in [17], known as the relational model, which also defined a series of relational algebra and calculus for data manipulation. In practice, these operations can be implemented via the Structured Query Language (SQL). There are various Relational Database Management Systems (RDBMS) options for back-end development in Python, such as SQL Server, Oracle Database, PostgreSQL, MySQL, SQLite, etc.

The non-relational database, also referred to as a NoSQL (Not only SQL) database, does not require a predefined schema for data representation. Some sub-classes of this category include: Key-value, Graph, Document, and Object databases, etc. These databases are normally used for

unstructured contents, such as text, image, audio, and so on. The popular Non-Relational Database Management Systems (NRDBMS) options are MongoDB, CouchDB, Oracle NoSQL, Neo4J, etc.

An Object-Relational Mapper (ORM) is "a code library that automate the transfer of data stored in relational database tables into objects" [18]. With an ORM, the developer can interact with the RDBMS using Python, instead of working with SQL which may be tedious. In addition, the ORM makes it easy for communication between different relational databases. For example, a developer could use SQLite for local development and MySQL in production without changing any code. There are various ORM implementations written in Python, including SQLAlchemy, The Django ORM, SQLObject etc.

2.2 Cloud-based/Distributed Systems for Engineering Design

Despite the extensive utilization of cloud computing for business purposes, research is still ongoing for its application in the field of engineering design, especially for aerospace developments. Existing literature can be categorized into three groups: cloud-based Computer Aid Design (CAD) systems, cloud-based High-Performance Computing (HPC) systems, general cloud-based design methodologies and systems for integration of tools/models.

2.2.1 Cloud-based Computer Aid Design (CAD) systems

Creation of component geometry plays an important role in (especially mechanical) engineering design. Therefore, much research has been devoted to development of cloud-based CAD systems [19–24]. These systems are commonly implemented as web-based applications using the client-server architecture, where the client provides a user interface while the actual CAD software and other functional components are installed on a server [22].

In [19,20], multiple servers are used for web portal, application hosing, coordination, data storage, Graphics Processing Unit (GPU) rendering, and scientific calculation, respectively. The user is first connected to the web portal server, then redirected by the coordinator server to the requested CAD tools are installed on another machine. The data, GPU, and calculation server will provide relevant computing resources during the application execution., A Product Lifecycle Management (PLM) method enabling collaboration between legacy (native applications) users and cloud-based users is described in [21]. The process involves definition of a Unified Product Structure (UPS) and corresponding requirements over the cloud, while legacy users can add/modify parts at the authority of the administrator. This case study was implemented with Dassault Systemes 3D Experience [25] (representing the cloud platform) and CATIA V5 [26] (native application). Reported in [23,24] are several prototypes for multiple users to make changes simultaneously on a single part file implemented in a server. These prototypes were implemented by integrating some existing CAx software API libraries and additional Client-Server plugins. Methods were also proposed to decompose a single component geometry so that different designers can work on the same component.

2.2.2 Cloud-based High-Performance Computing (HPC) systems

Development of cloud-based HPC systems [27–31] are mainly motivated by high-fidelity and large-scale simulations, such as Computational Fluid Dynamics (CFD) simulation of entire aircraft. These studies are computationally intensive, which demand hi-spec hardware resources such as supercomputer clusters. However, constructing and maintaining these supercomputer infrastructures is difficult and expensive, which requires remote sharing of HPC resources.

In general, cloud-based HPC is achieved with virtualization technology, so that hi-spec hardware resources (along with simulation software) can be packaged and accessed remotely. For instance, Ren et. al. [29] developed virtualization-based simulation platform (VSIM) to support multidisciplinary design of complex products. This platform is composed of simulation resources layer, virtualization layer, simulation services layer, and simulation application layer. The simulation resources layer contains all the hardware (CPU, memory, data storage, etc.) and software (simulation tools, models, knowledge base, etc.) resources. The virtualization layer maps these resources into virtual machines templates, and manages them as a pool for calling from different domain-specific simulation. The simulation services layer encapsulates simulation functions into services under a SOA framework. It also contains modules for scheduling, monitoring, deployment of all the services, etc. Finally, the application layer provides the user interfaces and web portal as an integrated design environment. This platform was applied for virtual prototyping of an aircraft undercarriage system, which involves simulation models of aerodynamics, mechanical systems, hydraulic systems, and multi-body dynamics.

Within this context, an induced problem is to optimize task schedule and resources allocation. Peng

et. al. [31] proposed a knowledge-based approach, in which a radial basis function neural network is trained to calculate the resource requirements (e.g., CPU frequency, disk size, etc.) according to the specification of a simulation task (e.g., software type, tenant size, etc.).

2.2.3 General cloud-based design methodologies and systems

Schaefer et. al. [32] proposed a conceptual model of Cloud-Based Design and Manufacture (CBDM), where the design and manufacture tasks are considered as services and delivered to different consumers in a crowd-sourcing paradigm. This process, as further defined by Wu et. al. [33], utilizes a search engine to receive requests for quotes (RFQs) from a consumer, then returns a list of candidate service providers, along with information such as prices, lead times, and quality level. The actual design process is then performed with Platform as a Service (PaaS) and Software as a Service (SaaS) tools, while the results (CAD models) are passed to on-demand manufacture websites such as 3dsystems.com [34].

A private cloud-based design environment, DMCloud, was developed as a prototype for demonstration and education purposes [35]. It employs the Moodle learning management system [36] as a centralized interfacing server (CIS) platform, which provides web-portals to CATIA V6 [26] and remote access to 3D printers and CNC milling machines, etc. It also integrates Google Doc [37] for information sharing and social networking tools such as chat rooms, forums, video conferences for design communication.

Similar concepts can be found in the research of Virtual Enterprise (VE), which is defined as “a temporary relationship with two or more participants… formed, operated, and dissolved to accomplish specific short term goals” [38].

While extensive research has been focused on the VE conceptual models [39,40], of particular interest to this research is the IT infrastructure which enables the design collaboration. For instance, in the VIVACE project [41], a Virtual Enterprise Collaboration Hub (VEC-Hub) was developed for information sharing and design integration between different project partners [42,43]. Within the context, local design systems are wrapped into web services and exposed to the VEC-Hub. The top level contains the portal services, which include management of access, security, and control over other services. These services are employed as user-interfaces to connect the hub with different partners. The core services are used to share product information, organizational roles, workflow, and reference data. Specifically, the product information includes, part, structures, documents, change proposals, and notifications, which are managed through PLCS /PLM web Services. The organizational roles are used for administrative purposes to ensure that resources are shared only between dedicated users. Workflow related services are used to create, edit, delete, start, monitor, and stop a remote workflow process. Finally, the reference data provide additional information on internal and external resources. A set of supplementary services are provided to handle request outside the core services. At the bottom of this architecture, is the data and infrastructure layer, which contains the networks, applications, and hardware for storing data and executing services. Communications between the infrastructure and services is through a messaging middleware layer, which employs Simple Object Access Protocol (SOAP) for transferring XML-based massages.

A cloud-based platform for automatic workflow generation [44,45] was developed in the NASA Europa spacecraft project [46]. The platform consists of three main repositories: Modeling Management System (MMS), Mission Planning and Sequencing (MPS), and ModelCenter Cloud (MCC). The MMS stores the system model of the spacecraft, which defines its structure, requirements, behaviors, and some key parameters. The system model is defined in Systems Modeling Language (SysML) [47], created by dedicated system modelers using MagicDraw [48] and then synchronized with the MMS repository. The MPS repository stores time-dependent data for mission design. The MCC is part of the ModelCenter software [49]. This repository stores simulation files and results, and is connected to a workflow execution server. The simulation models are created in parallel by domain experts and then published to the MCC repository. The conversion from SysML to executable workflow is achieved by ModelCenter MBSEPak, which maps the SysML parametric diagrams to corresponding simulation models [50]. JSON and REST APIs are used for communication between different repositories, and between local and cloud applications.

One recent advance in distributed design collaboration is the so-called AGILE paradigm [51–55]. This paradigm adopts the service-oriented architecture, where an engineering service is defined as: “a generically applicable software routine within an engineering domain, capable of automated handling of input and output data in a standardized format, which can be approached by other services via standard web or network technologies and ideally allows for batch execution without

requiring any intervention of the user" [55].

The implementation of the paradigm involves a set of specific participants and software, where the latter can be divided into two groups, namely the knowledge architecture [53] and the collaborative architecture [52].

Knowledge architecture is mainly used for definition of the design problem and construction of the workflow. For instance, a commercial web-based application named KE-chain [56], serves as a project management platform for hosting information of all the agents (participants) and design competences (models and tools). KADMOS is used for MDO formulation based on graph theory [57]. VISTOMS [58] is used for visualization of the MDO system. The output is an inexecutable "blueprint" of the automatic design process (workflow), which concludes the application of the knowledge architecture.

The collaborative architecture handles the actual execution of the workflow and studies. This is achieved with two alternative applications: Optimus and RCE. The former is a commercial software, which also contains modules for various analysis and post-processing functions [59]. The latter is an open-source software with similar functionalities [60–62]. The RCE also allows hosting of engineering services on various dedicated servers connected by a relay server for remote accessing. As the workflow contains both local and remote design tools and models, a client-server process is applied. This process uses a technology named Brics, which contains protocols and middleware for notification of remote engineers and the exchange of data [63].

During the process, there are two software-independent schemas for data storage and exchange. The first one is a product schema called CPACS, which is a structured parametric model to represent the aircraft [64,65]. The second is a workflow schema named CMDOWS, which stores the non-executable problem formulation (e.g. the model connections, MDO solution strategy, etc.) [66]. Both schemas are open-sourced and based on Extensible Markup Language (XML).

2.3 Discussion

Following an extensive review of cloud computing technology and its recent applications in the field of engineering design it could be concluded that SOA is widely adopted in academic and industrial applications. Two common features could be identified: 1) distributed models and tools are considered as engineering services; 2) the remote execution of a model/tool follows the client-server paradigm. In general, many efforts have been devoted to CAD and HPC systems, while research on workflow orchestration and distributed design environment is still an area worth further exploration. Specifically:

- There is still a lack of an integrated tool for orchestrating the workflow within the distributed design environment. For instance, the AIGILE paradigm relies on a collection of software and plug-ins to achieve the purposes. Although this setup provides more flexibility in configuring the design problem, it also leads to a complicated process, where specific integrators and collaborative engineers are required to perform the orchestration task.
- The existing methods and tools have been focused on the connections between models, while less attention has been paid to the interactions between designers. However, the computational workflow should be more than just a push-button process. For instance, expert experience is very important in the setup of many design models and tools, and also for the inspection of the computational results. In addition, the designer should be involved when there is a need to perform a trade-off study, not necessarily at the end of the workflow execution, but also during the process, within each sub-design problem.
- From implementation perspective, it is still not clear what would be the pros and cons of involving commercial-off-the-shelf cloud providers in the development of collaborative design environments. Using a third-party cloud bring security issues. However, it also provides a more flexible and cost-effective solution than deploying on-premises servers to host and execute design workflows.

3. Proposed Approach

3.1 Requirements Specification

Following an extensive literature review summarized in the previous section and a dialogue with the industrial partners in the COLIBRI project, several major requirements for cloud-based collaborative (computational) design have been identified:

- **Remote execution**: The cloud-based environment should facilitate remote model executions.

- **Security**: To protect intellectual property and external access to computational resources, the original developer should be able to deny irrelevant requests and be able to decide whether the model is stored locally or in the cloud.
- **Automatic sequencing**: The collaborative design environment should be able to schedule all the design tasks in a proper order, based on the model (inter) dependencies, e.g., sequentially, iteratively, or in parallel.
- **Automatic data exchange**: Based on the dependencies, the computational results of some models and tools should be collected and fed to the others automatically, so that the computation is not interrupted. However, the designer should be able to oversee the process and halt the execution if needed.
- **Dynamic creation of design studies**: Given a computational workflow (see 'Automatic sequencing' above), various studies can be dynamically created by the employment of cloud-based numerical treatments (methods), including: Design of Experiments (DoE), optimization, trade-off and sensitivity analysis, uncertainty, etc.
- **Visualization and Data Analytics**: The computational results should be presented to all the relevant designers for decision making. This requires production of manipulable (interactive) plots and rapid CAD models for interactive design space exploration and trade-off studies.
- **Design coordination**: The cloud-based environment should also be used to enhance interactions between different designers/design teams, so that the designer is always kept in the loop.

3.2 Architecture Overview

To implement and validate the requirements specified above, a cloud-base prototype application named "AirCADia Nebos" is developed in this research. This application adopts a microservice architecture, as illustrated in Figure 1, where all the components are implemented as microservices. These are loosely coupled and can be remotely accessible on the cloud-based environment. The back-end of the application (left side of Figure 1) consists of three repositories to host computational models, functional modules, and databases, respectively. The front-end (on the right side of the figure) includes several Graphical User Interfaces (GUI), which are used for invoking the back-end components and can be accessed thought web browsers. To protect intellectual properties, a security layer is developed for authentication and authorization, which grants control to different users, depending on their roles in the design process. Based on the discussion with industrial partners, the microservices are accessible via REST API's and the communication is based on JSON files. It is also decided that Microsoft Azure Cloud [67] is used for hosting the prototype.

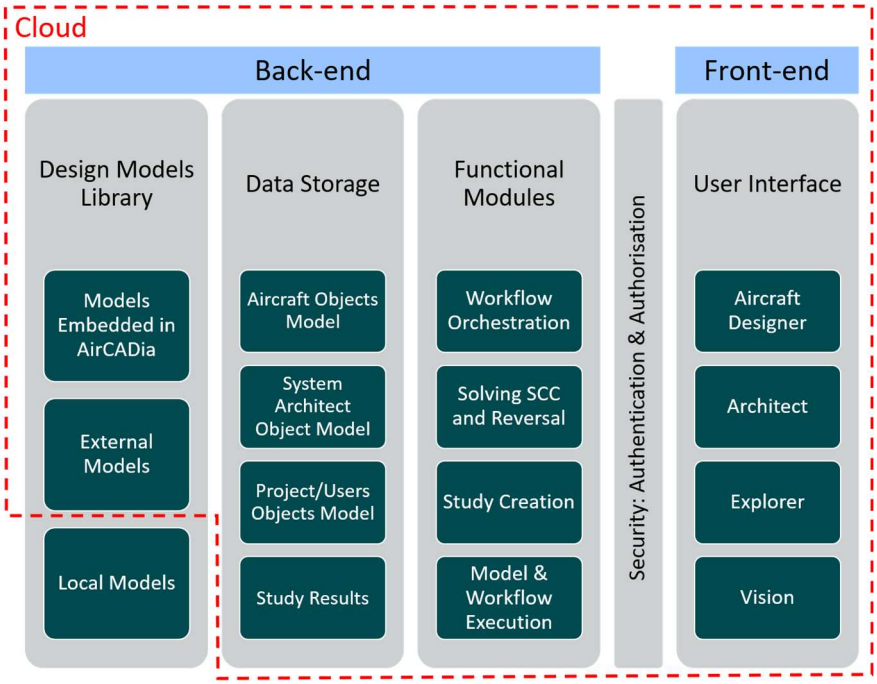

Figure 1 – AirCADia Nebos Architecture

3.3 Model Library

The computational model refers to the models used for design and analysis, such as empirical equations and simulations, e.g., CFD, FEM.

Depending on the owners, the models can be classified into three categories. The first category includes models which are created by the academic research group and embedded in AirCADia Nebos. These models are accessible for all the users of this cloud application. The second category includes external models adopted from the open-source domain or provided by the industrial/research partners who are willing to upload their models on the cloud repository. These models are shared among specific partners within a project. For security purposes, some partners may prefer to host their models only on local domains. These models form the third category, which will be accessed by implementing the local host as a server. This part is still under development.

Apart from the third category, models in the first two categories are hosted on the Microsoft Azure Cloud. The remote execution process is illustrated in Figure 2. Currently, the models are mainly coded in C# and python. The former is implemented using ASP.NET core web application framework, while the latter employs the FLASK framework. If the original model is provided as an executable file (.exe), a C# wrapper will be created to write/read the input/output files.

The model repository is defined by a Universal Resource Identifier (URL). To access the model, an HTTP POST request is sent to this URL, from a client or another microservice. The request body is a JSON file which specifies the model name, values of input variables, and expected output variables. If the execution is successful, a response will be returned as a JSON file, with updated values for the output variables.

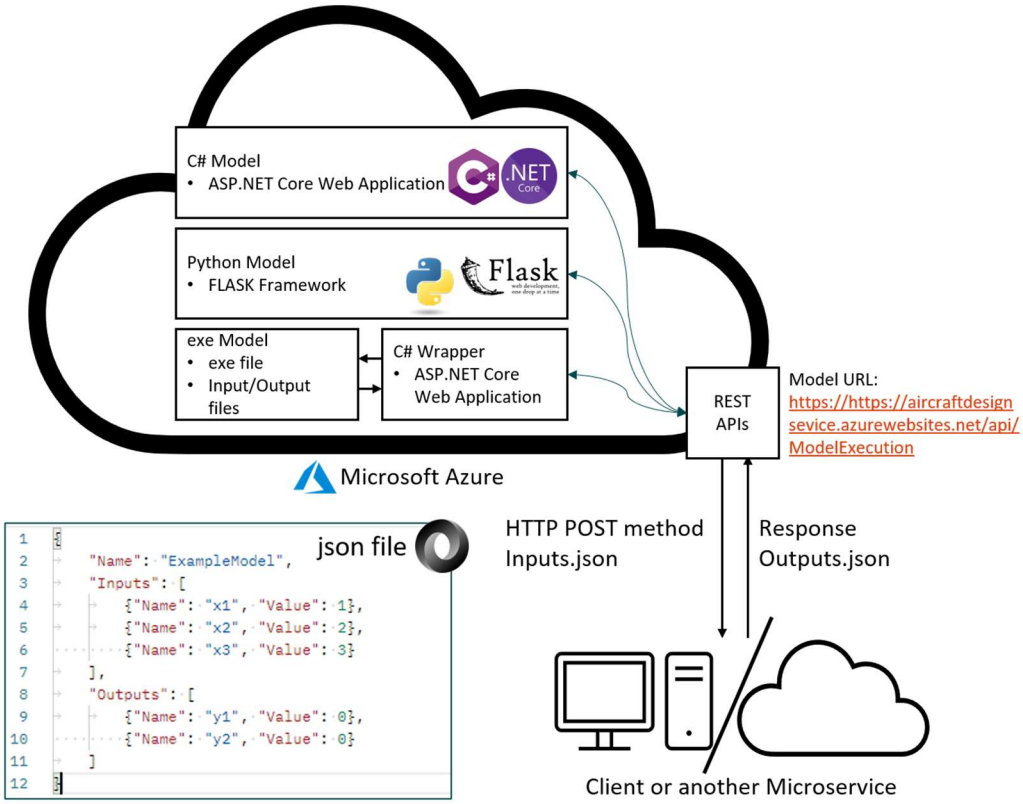

Figure 2 – Remote model execution on the cloud

3.4 Data Storage

The data stored on the cloud are encapsulated in four parts: aircraft, aircraft systems, project, and study results. The first three parts are represented as object models in JSON format, while the last is stored as SQL databases.

The aircraft object model defines the geometry, mission profile, and general performance of the aircraft under design. That is, it can be regarded as a “universal” representation for early-stage aircraft design. The geometry parameterization [68] is based on the Class Shape Transformation (CST) method [69]. The mission section includes distance, altitude, and Mach number for take-off,

climb, cruise, decent, and landing segments, respectively. The performance section includes top-level performance variables such as take-off/landing field length, time to climb, fuel consumption, weight decomposition, etc.

The system object model defines the aircraft system architecture using the Requirements, Functional, Logical, Physical (RFLP) approach. Further details of this definition could be found in [70].

The project object model defines the formulation of the design problem, which contains four major elements: data, model, subprocess, and study. As illustrated in Figure 3, at the bottom level of the hierarchy is the Data object, which can be used to represent any type of variables in a computation (e.g., design variables, constants, intermediate results, performance variables, etc.). The Model object is a basic executable component, which defines the computational relationship between a set of input and output variables. The assembly of different models defines a computational workflow, implemented as a Subprocess object. Within a subprocess, the models are executed in a specific sequence, and the results are passed from upstream models to the downstream ones automatically. Finally, Study object can be created for Design of Experiments (DoE), optimization, uncertainty propagation, and sensitivity analysis. It consists of a model/subprocess and certain numerical methods (finite differencing, fix point iteration, genetic algorithm, etc.), where the latter are implemented as Treatment objects. Further details of this definition could be found in [71,72].

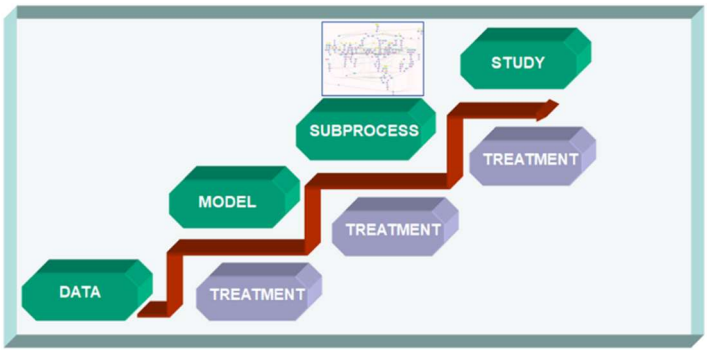

Figure 3 – Schematic view of AirCADia object hierarchy.

In general, the study results produced in design computation are well-structured. For instance, the results of a DoE study contain a list of design points, each of which is defined by a set of design variables, constants, and performances outputs. It is therefore decided to employ relational database management system (RDBMS) in the collaborative design environment.

Several options were considered. MySQL is one of the most widely used RDBMS, due to its simplicity and open source. PostgreSQL is also an open-source system and has object-oriented features to handle more complex data structures. SQLite is built into Python, but needs to be stored in a single file on local disk. However, it is very easy to use and test within the local environment. Therefore, it has been decided to use SQLite during the development and PostgreSQL/MySQL will be used for the actual cloud application.

To connect the numeric results with the object models, an Object-Relational Mapper (ORM) is used. Among the current solutions, SQLAlchemy [73] is a popular solution, due to its effective connection/pooling infrastructure for complex queries and mappings. It also has good compatibilities with PostgreSQL, MySQL, and SQLite, which are chosen for the collaborative design environment. Therefore, SQLAlchemy will be employed in the current research.

3.5 Functional Modules

The functional modules are used for manipulating the executable components, such as models, workflows, and studies. These modules are based on existing enablers for model-based design and system engineering developed in Cranfield university during previous research projects [41,74–76].

To orchestrate a workflow, the user can specify which models should be included. Algorithms were developed to enable automatic assembly of the selected models into a computational workflow, where the outputs of one model will be used as the inputs of others, based on their dependencies [77,78]. This capability was further developed in [79], where methods and algorithms were proposed to automatically convert a system architecture into a computational workflow. It allows the designer to perform on-the-fly analysis while defining the system architecture of a complex product.

In case of coupled models, a Strongly Connected Component (SCC) will be identified, and a solver will be set up to converge the iteration. This process is also automated behind the scenes. Another capability is to reverse a model or a workflow, by swapping its original input and output variables. This is achieved by in-built solvers and it allows the designer to ask “what-if” questions and solve inverse design problems. For instance, given a specific value of a performance output, what should be the value of a design variable. The reader is referred to [80,81] for further details.

The study creation module creates study objects as outlined in Section 3.4. For each workflow or study, a specific execution strategy will be produced in the background. The execution module sends out requests (as described in Section 3.3) to all the relevant models according to the execution strategy.

3.6 User Interface

The user interfaces are developed as webpages which can be accessed with standard web browsers. These webpages are written in HTML and Javascript for dynamic interaction with the designer. Third-party libraries are used, including Webix for UI design [82], D3.js for data visualization [83], and Three.js for 3D geometry [84]. Currently the cloud version of the Architect module is under development. The Aircraft Designer, Explorer, and Vision are presented in this section.

3.6.1 Interactive Aircraft Definition

The aircraft designer interface is used for interactive definition of the aircraft geometry and mission profile. As discussed in Section 3.4, the aircraft is fully parameterized. For each major component, such as the wing, fuselage, horizontal and vertical tails, etc., a form is used for specifying its parameters, as illustrated in Figure 4, left. The geometry can be updated simultaneously as the input values are changed. The mission profile is defined in a similar way, with another form as shown in Figure 4, right.

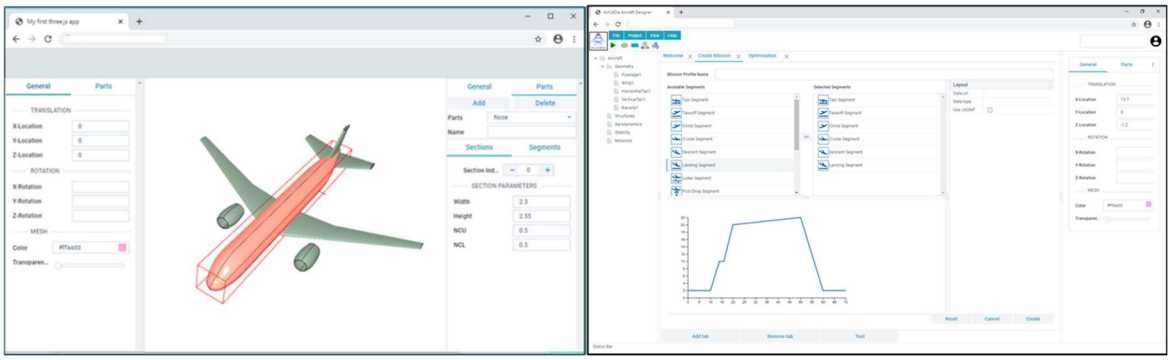

Figure 4 – Aircraft design interface

The definition of the aircraft is saved as an object model in a JSON file. The designer can assign a specific computational model (from the model library) to the aircraft or one of its components. This link between aircraft object and computational models is also saved in the JSON file. To perform a design analysis (as illustrated in Figure 4), the aircraft designer interface will first load the aircraft geometry and the associated model names from the aircraft object JSON file. Then for each model to be executed, a separate input JSON file will be produced, which contains the model name, and values of input variables, as shown in Figure 5. This JSON file will be send to the model repository as a request using the HTTP POST method. The results will be sent back as an output JSON, which will be used to update the aircraft geometry and performance in the interface. The changes will also be saved in the aircraft object JSON file.

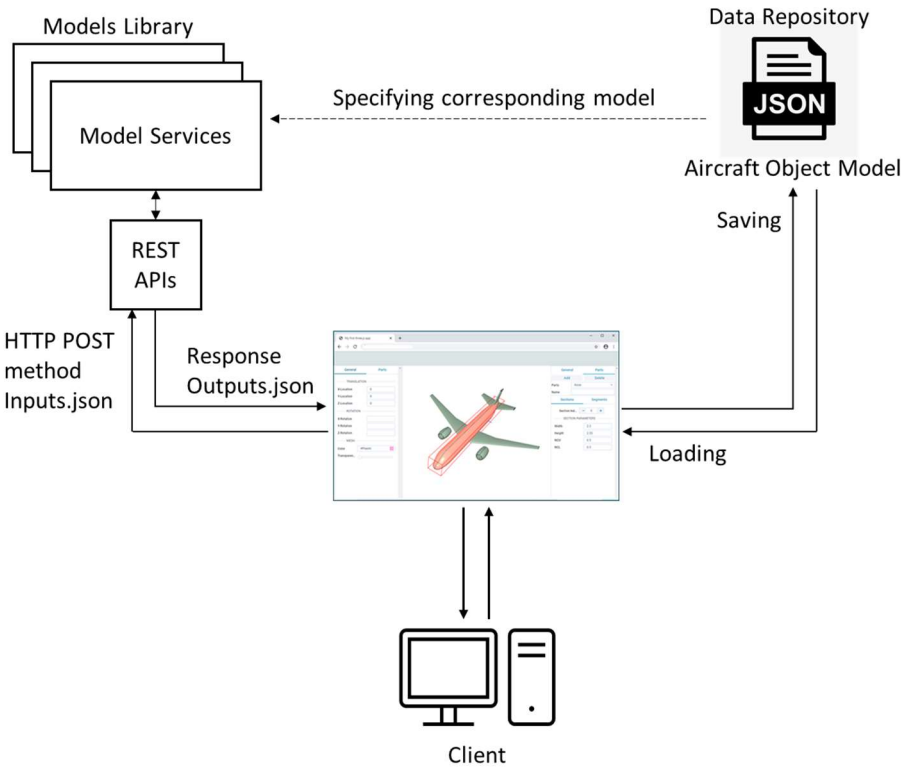

Figure 5 – Model execution via the aircraft designer interface

3.6.2 Explorer and Vision

While the aircraft designer interface is used specifically for aircraft design, the explorer interface can be used for more general problems. This interface (Figure 6 Left) mainly handles the project object model, as discussed in Section 3.4. The designer can create, modify, and delete models, workflows, and studies. During these operations, the functional module as discussed in Section 3.5 will be invoked, (e.g. to connect the models as a workflow). The project object will also be saved as a JSON file in the data repository.

After execution of a workflow or study, the computational results are stored in the cloud and need to be presented to all the relevant designers for decision-making. This can be achieved by the Vison interface as shown in Figure 6 right. It facilitates a number of interactive plots (e.g. scatter plots, parallel coordinates plot, surface plots, etc.) which are synchronized together, so that moving a design point in one plot will be reflected simultaneously across all the other plots in the workspace.

These plots are also linked to the computational workflow for potential additional execution, which enables the designer to test “what if” scenarios for further exploration of the design space and to conduct trade-off between the performance variables. Data analytics could also be used to extract design rules, for instance, by conducting correlation and sensitivity studies on different variables. This helps to capture implicit relationship between different variables and construct a knowledge base for future applications.

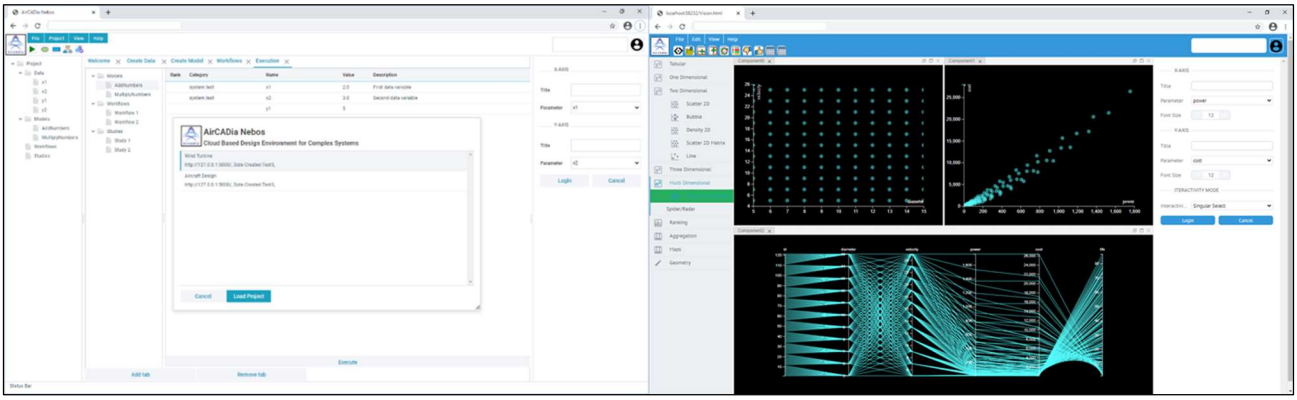

Figure 6 – Explorer and Vision Interface

4. Evaluation

For demonstration and evaluation, two use-cases are conducted with the cloud-based design environment described above. The first use-case is an aircraft sizing study, which is used to test the cloud-based workflow orchestration capability. The second one is an airframe-engine matching case study to demonstrate collaborative design optimization between different (distributed) design teams. This use-case was initially applied in the COLIBRI's predecessor project, APROCONE [74], to demonstrate design coordination, but was based on a monolithic workflow setup [85]. In this evaluation, the models and executions are migrated to the cloud, and the designers will use different computers to perform the study.

4.1 Aircraft Sizing Workflow Orchestration

Early-stage aircraft sizing involves a large number of computational models (e.g. empirical equations, lookup tables, low fidelity tools, etc.), whose input and output variables should be connected to each other during the computation. Figure 7 (a) shows the web interface for workflow orchestration, where the box on the left lists all the models available in library, while the box on the right contains the selected ones to be connected. After confirming the selection, the cloud-based orchestration module enables automatic assembly and sequencing of these models, based on the variable dependencies. The completed workflow is identical to the one produced by the local version of AirCADia, and is illustrated in Figure 7 (b).

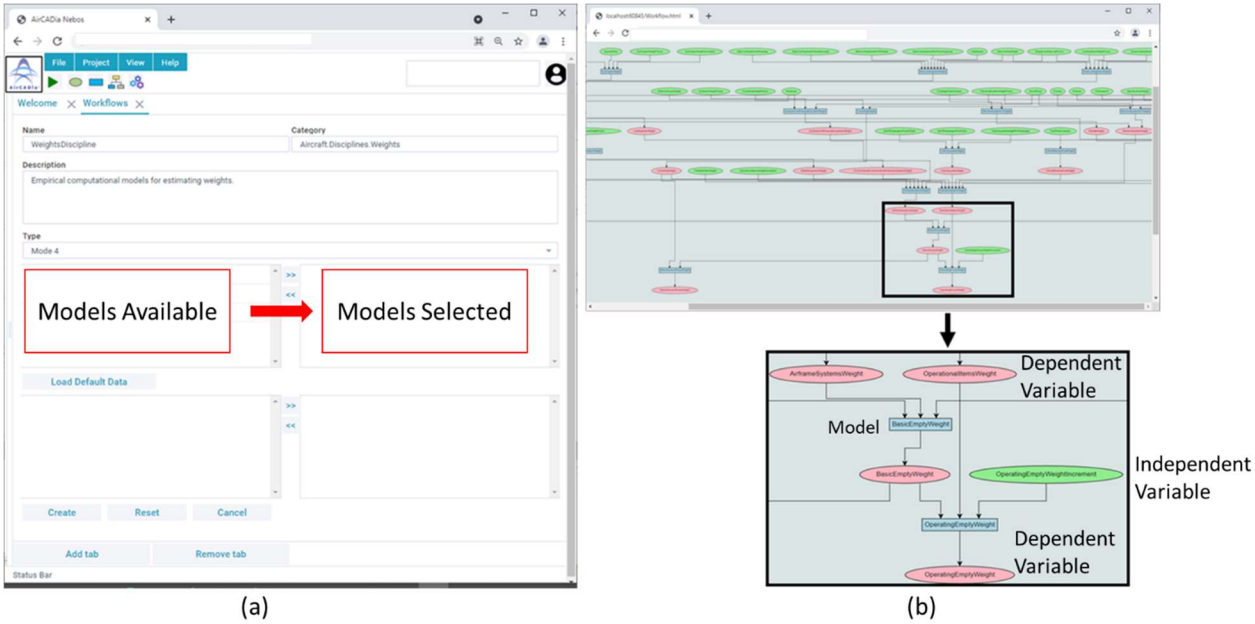

Figure 7 – Automatic workflow orchestration

4.2 Airframe and Engine Matching

As illustrate in Figure 8, the airframers first employ conceptual aircraft design tools to identify engine

thrust requirements. The latter are then passed on to the engine manufacturer as specifications. The engine designers apply their cycle analysis tools to generate an engine performance deck, which is then sent back to the airframe designers for assessment. The new engine performance deck will result in a modification of the airframe and thus will lead to revised thrust requirements. Therefore, an iterative process will be applied to update the airframe and engine, until a convergence is achieved.

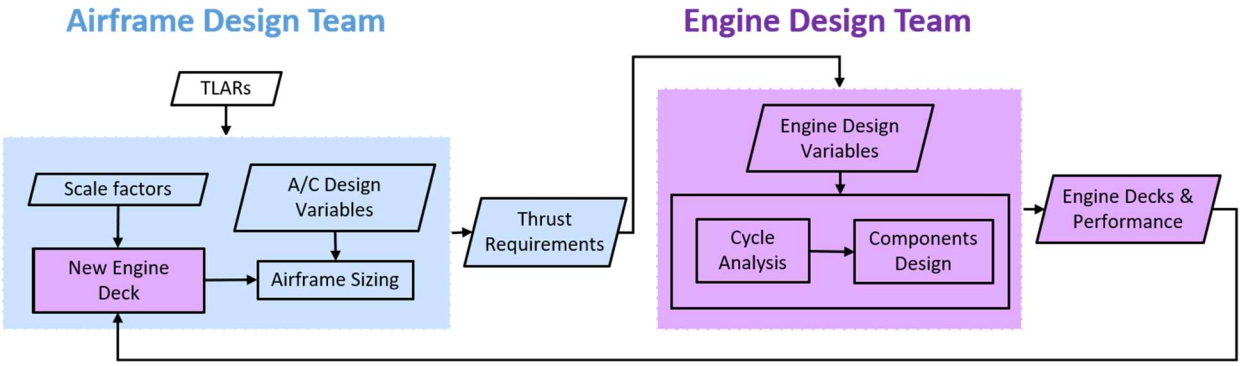

Figure 8 – Airframe and engine matching use-case

In the original setup, FLOPS [86] was used for the airframe design while NPSS [87] was used for the engine cycle analysis. However, NPSS is locked to a physical IP of a local workstation, therefore it cannot be implemented as a microservice over the cloud. As mentioned in Section 3.3, the remote accessing of local models is still under development. In this demonstration, the FLOPS engine module is used as a sperate model to temporarily replace the NPSS model for producing the engine deck. At the same time, cycle analysis is disabled in the main program of FLOPS, so that the latter is isolated from the engine module.

An A320 like aircraft is chosen as an illustrative example, whereas the problem setups for the airframer and engine designer are shown in Table 1 and Table 2, respectively. The airframe optimization is based on a rubberized engine, with three scaling factors for maximum take-off, maximum climb, and maximum cruise thrust, respectively. A baseline engine deck is used in the first iteration. From the second iteration, it is replaced with a new engine deck from the cycle analysis (conducted by the engine design team). It should be noted that this use-case is to demonstrate the cloud-based design environment rather than the actual collaborative design case study. The latter will be further developed in the COLIBRI project.

Table 1 – Airframe Design Problem Setup

Category	Variable	Symbol	Value
Design Variables	Wing Area	S_W	$[1200, 1400]\ ft^2$
	Aspect Ratio	AR	$[8, 12]$
	Taper Ratio	TR	$[0.2, 0.3]$
	0.25 Chord Line Sweep Angle	$\Lambda_{0.25}$	$[25, 30]\ degree$
	Engine Deck	N/A	$Baseline/From\ Engine\ Team$
	Engine Scaling Maximum Take-Off	k_{MTO}	$[0.95, 1.05]$
	Engine Scaling Maximum Climb	k_{MCL}	$[0.95, 1.05]$
	Engine Scaling Maximum Cruise	k_{MCR}	$[0.95, 1.05]$
Parameters	Cruise Altitude	H_{Cr}	$37000\ ft$
	Cruise Mach	$Mach_{Cr}$	0.78
	Number of Passengers	N_{Pax}	160
	Design Range	R	3500
Constraints	Take-off Field Length	$TOFL$	$\leq 7000\ ft$
	Landing Field Length	LFL	$\leq 6000\ ft$
	Design Range	R_{Des}	$\geq 3300\ nm$

	Approach Velocity	V_{APP}	$\leq 130\ kts$
	Time to Climb	TTC	$minimise$
Objective Function	Block Fuel	W_F	$minimise$
	Time to Climb	TTC	$minimise$

Table 2 – Engine Design Problem Setup

Category	Variable	Symbol	Value
Design Variables	Sea Level Static Thrust	$SLST$	$[20000, 40000]\ lbf$
	Bypass Ratio	BPR	$[5, 7]$
	Fan Pressure Ratio	FPR	$[1.6, 1.7]$
	Overall Pressure Ratio	OPR	$[27, 30]$
Parameters	Cruise Altitude	H_{Cr}	$37000\ ft$
	Cruise Mach	$Mach_{Cr}$	0.78
Constraints	End of Runway Thrust	T_{EoR}	$\geq Req\ From\ Airframe\ Team$
	Top of Climb Thrust	T_{ToC}	$\geq Req\ From\ Airframe\ Team$
	Mid-Cruise Thrust	T_{MCR}	$\geq Req\ From\ Airframe\ Team$
Objective Function	Specific Fuel Consumption	SFC	$minimise$

4.3 Implementation and Results

The implementation of the use-case in the cloud-based environment is illustrated in Figure 9. The airframe designer will first formulate the design problem with the Explorer interface, by specifying the design variables, constraints, and objective functions. During this process, the Explorer interface will invoke the study creation microservice, as indicated by step ① in Figure 9. The project object model will be saved in the data repository, as shown by step ②. Meanwhile, the execution microservice will invoke the FLOPS main program repeatedly during the optimization process (step ③). The optimization results are also saved (step ④) in the data repository and loaded (step ⑤) by the Vision and Aircraft designer interface for decision making (as shown by Figure 10). After the airframe is frozen (in the current iteration), the thrust requirements are published (Figure 11) and passed on to the engine design team (step ⑥). Similar steps are followed to conduct the engine optimization (steps ⑦ to ⑫). The results and the engine deck are illustrated in Figure 12 and Figure 13, respectively. The new engine deck will be used for the next iteration of the airframe design optimization.

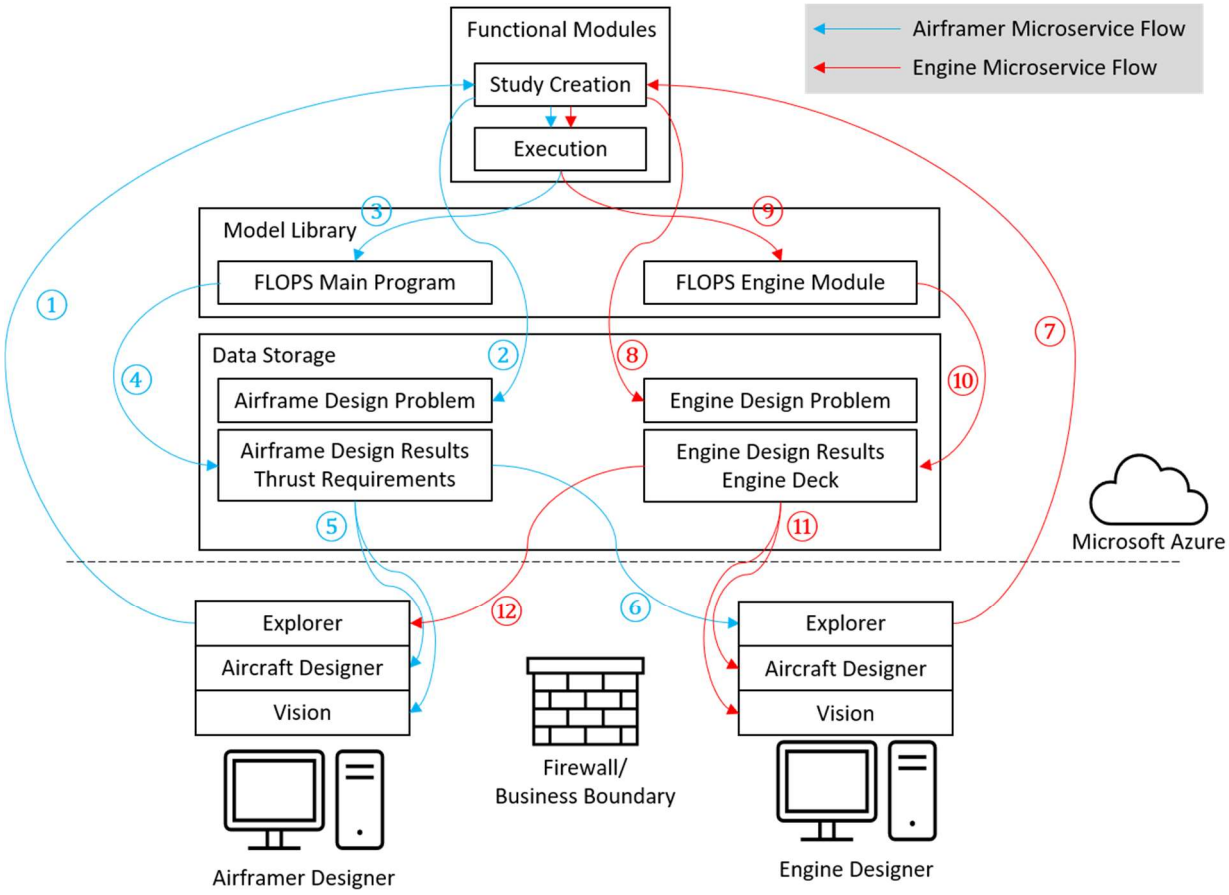

Figure 9 – Implementation of the use-case in the cloud-based environment

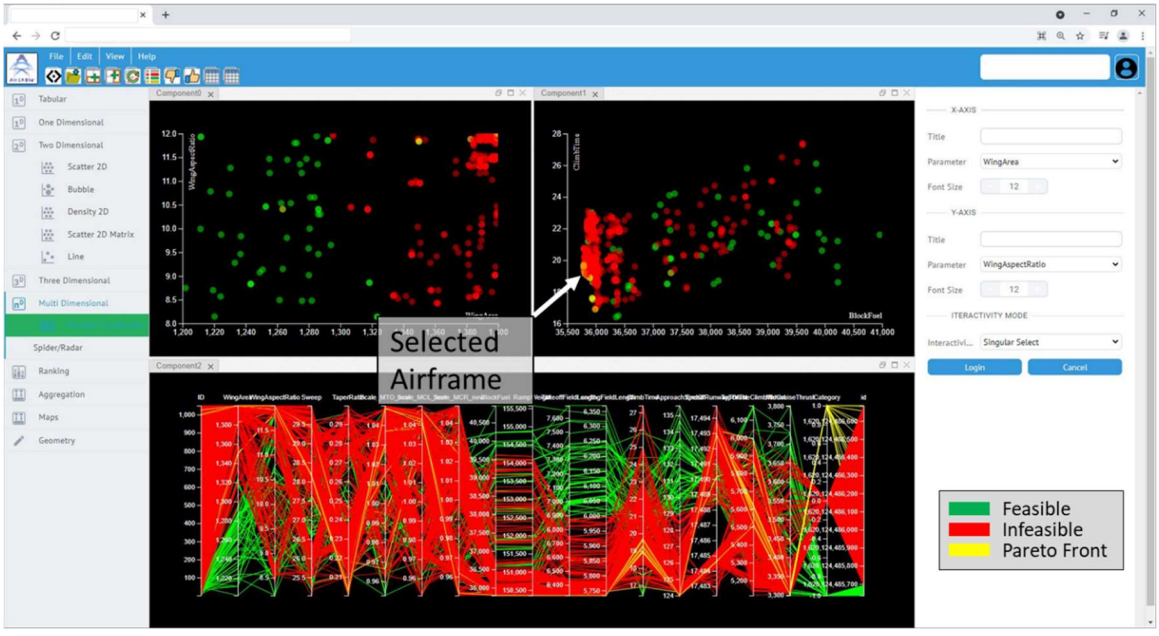

Figure 10 – Airframe optimization result in Vision Interface.

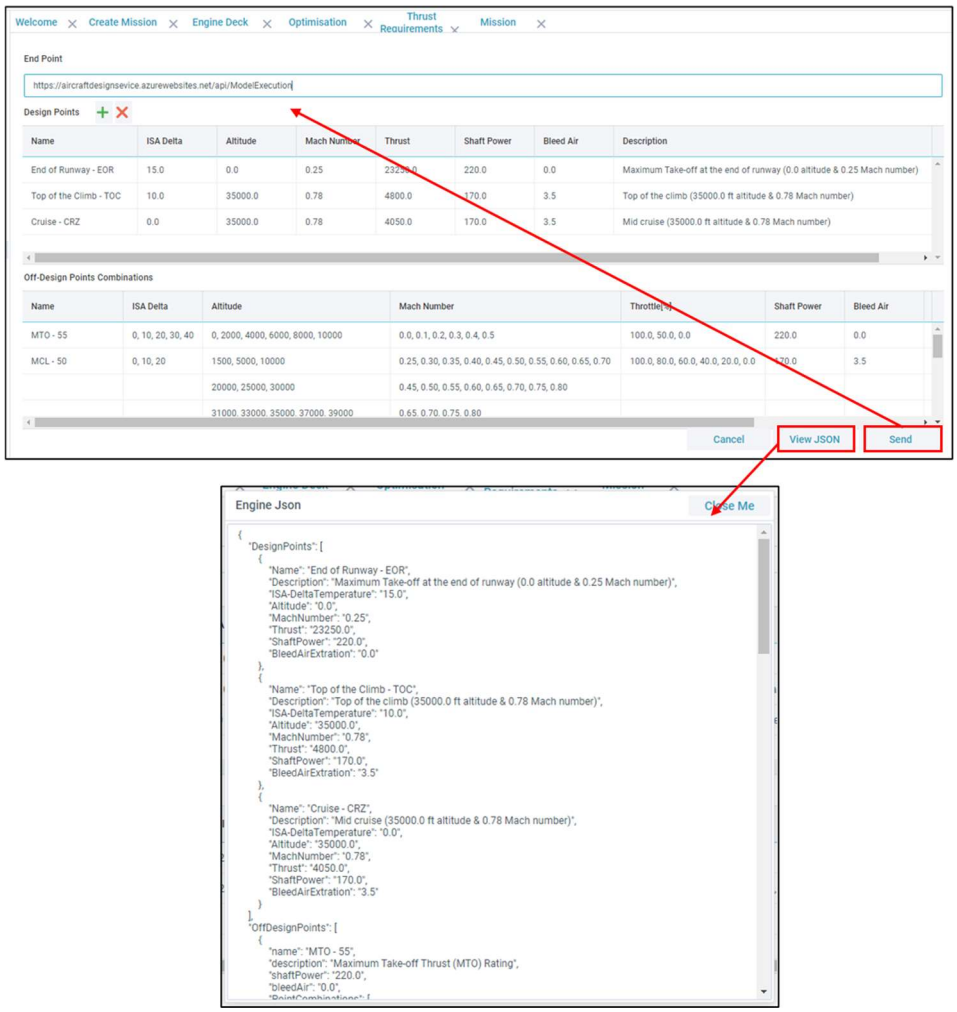

Figure 11 – Publishing thrust requirements.

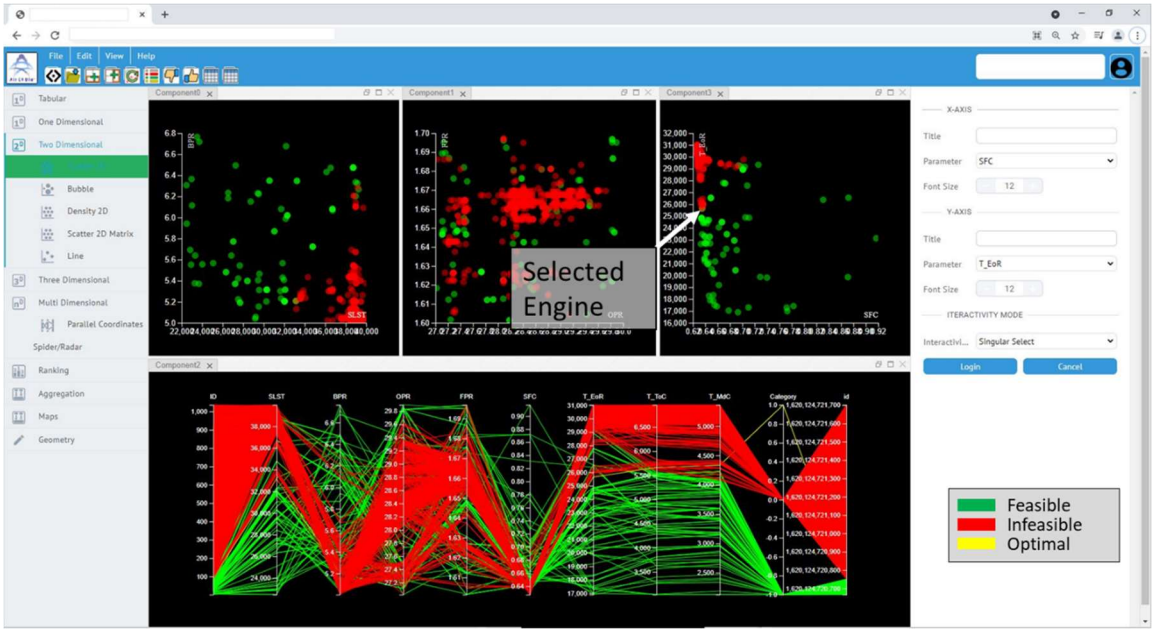

Figure 12 – Engine optimization result in Vision Interface.

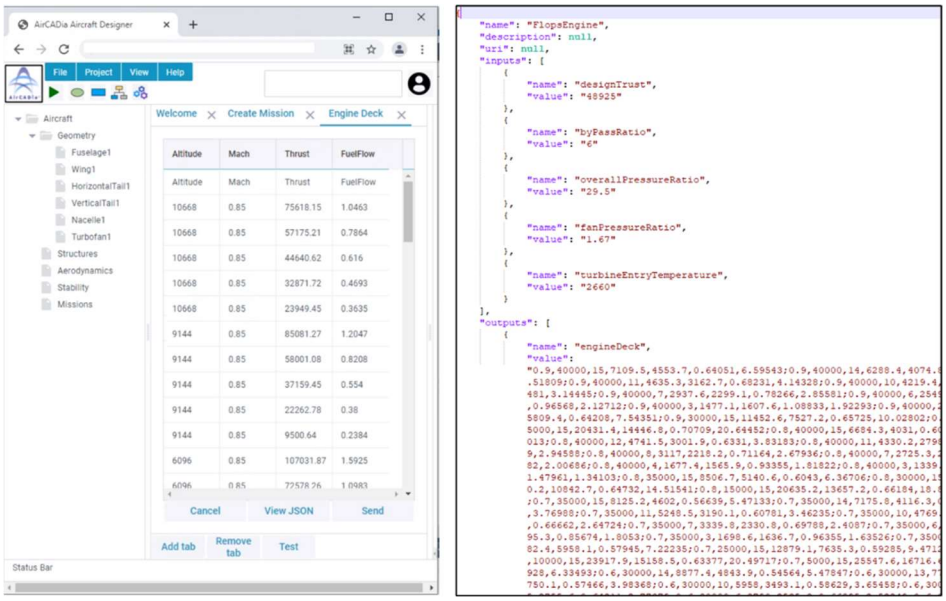

Figure 13 – Publishing engine deck

5. Summary and Conclusions

Presented in this paper is an outline specification for a cloud-based collaborative aircraft design environment.

Following an extensive literature review of cloud computing technology and its recent applications in the field of engineering design, it was found that while aspects of the service-oriented architecture approach have been widely adopted in academic and industrial applications, areas which require further exploration include:

- There is still a lack of an integrated tool for orchestrating the workflow within the distributed design environment.
- The existing methods and tools have been focused on the connections between models, while less attention has been paid to the interactions between designers.
- The pros and cons of relying on commercial-off-the-shelf cloud providers in the development of collaborative design environments in industry are still to be determined.

Several major requirements for cloud-based collaborative (computational) design have been identified from the state-of-the-art review and from discussions between Cranfield University and the industrial partners in the COLIBRI project, including: remote execution, security, automatic sequencing (workflow orchestration), automatic data exchange, dynamic creation of design studies, visualization and data analytics, and design coordination.

In order to validate the specified requirements, an architecture is proposed for exposing computational models and general engineering capabilities as microservices on the cloud. This architecture is implemented in a cloud-base prototype application, named “AirCADia Nebos”. Currently this prototype employs Microsoft Azure to host the microservices, while web-based user interfaces are implemented to invoke these capabilities and to visualize results over the cloud. The communication between microservices is based on JSON files and REST APIs.

Initial evaluation was performed with an aircraft sizing use-case to test the workflow orchestration capability and an airframe-engine matching use-case to demonstrate the remote collaboration capability. The latter was first implemented and ran on a local host computer as part COLIBRI’s predecessor project, APROCONE [74]. The results obtained from running the same computational models in the cloud-based environment via AirCADia Nebos are identical with those of the monolithic (non-distributed) implementation. Speed and reliability of the connection were not an explicit objective of the tests, but in the current test no problems were observed.

In conclusion, the proposed architecture and developed prototype can be used to share computational models and general engineering capabilities among distributed design teams, which enables collaborative optimization studies without installing any local instances of software and models. The results so far are promising and indicate that with the current problem scale, the actual

design solution will not be impacted by the migration to the cloud.

Future work will include further development of the prototype to handle the design coordination problem in a broader view, especially interactivity. This involves handling conflicting constraints, implicit interfaces, and evolving problem formulations and so forth. A comprehensive use-case will be developed in collaboration with the partners in the COLIBRI project, to test the scalability of the application. Last, but not least, to enhance security and intellectual property protection, an authentication and authorization mechanism will be developed for granting controls to different users, depending on their roles in the design process.

6. Acknowledgment

The research leading to these results has received funding from Innovate UK, under the Collaboration Across Business Boundaries (COLIBRI) project (Ref no. 113296).

Copyright Statement

7. References

[1] COLIBRI – Collaboration Across Business Boundaries. https://gtr.ukri.org/projects?ref=113296. Accessed Apr. 20, 2021.

[2] Laberis, B. *What Is the Cloud?*, O'Reilly Media, 2019.

[3] Mell, P. M., and Grance, T. *The NIST Definition of Cloud Computing*, NIST Special Publication 800-145, 2011.

[4] Web Services Architecture. https://www.w3.org/TR/ws-arch/#service. Accessed Apr. 23, 2021.

[5] SOA vs. Microservices: What's the Difference? | IBM. https://www.ibm.com/cloud/blog/soa-vs-microservices. Accessed Apr. 22, 2021.

[6] What Is SOA (Service-Oriented Architecture)? | IBM. https://www.ibm.com/cloud/learn/soa#toc-soa-exampl-ddeqU_3x. Accessed Apr. 22, 2021.

[7] MacKenzie, C.M., Laskey, K., McCabe, F., Brown, P.F., Metz, R. and Hamilton, B.A. *Reference Model for Service Oriented Architecture 1.0*, OASIS standard, 2006.

[8] Niknejad, N., Ismail, W., Ghani, I., Nazari, B., Bahari, M., and Hussin, A. R. B. C. "Understanding Service-Oriented Architecture (SOA): A Systematic Literature Review and Directions for Further Investigation." *Information Systems*, Vol. 91, 2020, p. 101491. https://doi.org/10.1016/j.is.2020.101491.

[9] Dragoni, N., Giallorenzo, S., Lafuente, A. L., Mazzara, M., Montesi, F., Mustafin, R., and Safina, L. Microservices: Yesterday, Today, and Tomorrow. In *Present and Ulterior Software Engineering*, Springer International Publishing, 2017, pp. 195–216.

[10] HTTP - Hypertext Transfer Protocol Overview. https://www.w3.org/Protocols/. Accessed Apr. 27, 2021.

[11] An Introduction to Remote Procedure Call. https://www.w3.org/History/1992/nfs_dxcern_mirror/rpc/doc/Introduction/Abstract.html. Accessed Jan. 27, 2021.

[12] Simple Object Access Protocol (SOAP) 1.1. https://www.w3.org/TR/2000/NOTE-SOAP-20000508/. Accessed Jan. 27, 2021.

[13] Fielding, R. *Architectural Styles and the Design of Network-Based Software Architectures*. Phd Thesis, University of California, 2000.

[14] Extensible Markup Language (XML). https://www.w3.org/XML/. Accessed Jan. 11, 2021.

[15] JSON. https://www.json.org/json-en.html. Accessed Apr. 23, 2021.

[16] Connolly, T., and Begg, C. *Database Systems: A Practical Approach to Design, Implementation and Management*. Pearson Education Limited, 2015.

[17] Codd, E. F. "A Relational Model of Data for Large Shared Data Banks." *Communications of the ACM*, Vol. 13, No. 6, 1970, pp. 377–387. https://doi.org/10.1145/362384.362685.

[18] Object-Relational Mappers (ORMs) - Full Stack Python. https://www.fullstackpython.com/object-relational-mappers-orms.html. Accessed Jan. 27, 2021.

[19] Andreadis, G., and Drossos, A. "A Cloud Based Framework for Automated CAD Design." *International Journal of Modern Manufacturing Technologies*, Vol. V, No. 2, 2013.

[20] Andreadis, G., Fourtounis, G., and Bouzakis, K. D. "Collaborative Design in the Era of Cloud Computing." *Advances in Engineering Software*, Vol. 81, No. C, 2015, pp. 66–72. https://doi.org/10.1016/j.advengsoft.2014.11.002.

[21] Sharma, S., Segonds, F., Maranzana, N., Chasset, D., and Frerebeau, V. Towards Cloud Based Collaborative Design – Analysis in Digital PLM Environment. In *IFIP International Conference on Product Lifecycle Management*, Turin, Italy, Vol. 540, pp. 261-270, 2018.

[22] Eynard, B., Liénard, S., Charles, S., and Odinot, A. "Web-Based Collaborative Engineering Support System: Applications in Mechanical Design and Structural Analysis." *Concurrent Engineering*, Vol. 13, No. 2, 2005, pp. 145–153. https://doi.org/10.1177/1063293X05053799.

[23] Red, E., French, D., Jensen, G., Walker, S. S., and Madsen, P. "Emerging Design Methods and Tools in Collaborative Product Development." *Journal of Computing and Information Science in Engineering*, Vol. 13, No. 3, 2013. https://doi.org/10.1115/1.4023917.

[24] Red, E., Jensen, G., Weerakoon, P., French, D., Benzley, S., and Merkley, K. "Architectural Limitations in Multi-User Computer-Aided Engineering Applications." *Computer and Information Science*, Vol. 6, No. 4, 2013, p. p1. https://doi.org/10.5539/cis.v6n4p1.

[25] The 3DEXPERIENCE Platform, a Game Changer for Business and Innovation | Dassault Systèmes. https://www.3ds.com/3dexperience. Accessed Apr. 27, 2021.

[26] CATIA™ Products - Dassault Systèmes®. https://www.3ds.com/products-services/catia/products/. Accessed Jan. 13, 2021.

[27] Zhimin, T., Qi, L., and Guangwen, Y. Integrating Cloud-Computing-Specific Model into Aircraft Design. In *IEEE International Conference on Cloud Computing*, No. 5931, pp. 632-637, 2009.

[28] Tian Z. M., L. Q. , Y. G. Application of Grid Computing in Aircraft Design. In *27th Congress of International Council of the Aeronautical Sciences*, Nice, France, ICAS2010-P2.5, 2010.

[29] Ren, L., Zhang, L., Tao, F., Zhang, X., Luo, Y., and Zhang, Y. "A Methodology towards Virtualisation-Based High Performance Simulation Platform Supporting Multidisciplinary Design of Complex Products." *Enterprise Information Systems*, Vol. 6, No. 3, 2012, pp. 267–290. https://doi.org/10.1080/17517575.2011.592598.

[30] Li, B. H., Chai, X., Hou, B., Zhu, W., Zhang, Y., Tang, Z., Song, C., and Mu, S. Cloud Simulation Platform. 2009.

[31] Peng, G., Wang, H., Dong, J., and Zhang, H. "Knowledge-Based Resource Allocation for Collaborative

Simulation Development in a Multi-Tenant Cloud Computing Environment." *IEEE Transactions on Services Computing*, Vol. 11, No. 2, 2018, pp. 306–317. https://doi.org/10.1109/TSC.2016.2518161.

[32] Schaefer, D., Thames, J. L., Wellman, R. D., Wu, D., and Rosen, D. W. "Distributed Collaborative Design and Manufacture in the Cloud-Motivation, Infrastructure, and Education." *Computers in Education Journal*, Vol. 22, No. 4, 2012, pp. 1–16.

[33] Wu, D., Rosen, D. W., Wang, L., and Schaefer, D. "Cloud-Based Design and Manufacturing: A New Paradigm in Digital Manufacturing and Design Innovation." *CAD Computer Aided Design*, Vol. 59, 2015, pp. 1–14. https://doi.org/10.1016/j.cad.2014.07.006.

[34] 3D Printers, Software, Manufacturing & Digital Healthcare | 3D Systems. https://www.3dsystems.com/. Accessed Dec. 9, 2020.

[35] Wu, D., Lane Thames, J., Rosen, D. W., and Schaefer, D. "Enhancing the Product Realization Process with Cloud-Based Design and Manufacturing Systems." *Journal of Computing and Information Science in Engineering*, Vol. 13, No. 4, 2013. https://doi.org/10.1115/1.4025257.

[36] Moodle: Online Learning with the World's Most Popular LMS. https://moodle.com/. Accessed Dec. 9, 2020.

[37] Google Docs: Free Online Documents for Personal Use. https://www.google.co.uk/docs/about/. Accessed Apr. 27, 2021.

[38] Reid, R. L., Rogers, K. J., Johnson, M. E., and Liles Donald H. Engineering the Virtual Enterprise. In *IERC Proceedings 1996, 5th Annual Industrial Engineering Research Conference*, pp. 485-490, 1996.

[39] Martinez, M. T., Fouletier, P., Park, K. H., and Favrel, J. "Virtual Enterprise - Organisation, Evolution and Control." *International Journal of Production Economics*, Vol. 74, No. 1–3, 2001, pp. 225–238. https://doi.org/10.1016/S0925-5273(01)00129-3.

[40] Esposito, E., and Evangelista, P. "Investigating Virtual Enterprise Models: Literature Review and Empirical Findings." *International Journal of Production Economics*, Vol. 148, 2014, pp. 145–157. https://doi.org/10.1016/j.ijpe.2013.10.003.

[41] Value Improvement through a Virtual Aeronautical Collaborative Enterprise (VIVACE). http://cordis.europa.eu/project/rcn/72825_en.html. Accessed Jun. 11, 2017.

[42] Wenzel, H., Almyren, F., Barner, J., Baalbergen, E., Meissner, B., and Lindeblad, M. Using the Virtual-Enterprise-Collaboration Hub for Distributed Engine Optimization. In *Advances in Collaborative Civil Aeronautical Multidisciplinary Design Optimization* (M. D. Guenov and E. Kesseler, eds.), American Institute of Aeronautics and Astronautics, Inc, 2009, pp. 331–363.

[43] Kesseler, E., and Guenov, M. D. *Advances in Collaborative Civil Aeronautical Multidisciplinary Design Optimization*. American Institute of Aeronautics and Astronautics, Reston ,VA, 2010.

[44] Post, E., Dinkel, K., Lee, E., Cole, B., Kim, H., and Nairouz, B. Cloud-Based Orchestration of a Model-Based Power and Data Analysis Toolchain. In *2016 IEEE Aerospace Conference*, MT, USA, pp. 1-12, 2016.

[45] Cole, B., and Simmons, J. An Integrated Systems Modeling and Analysis Platform for Flight Project Work. In *AIAA SPACE 2016*, California, USA, AIAA 2016-5471, p. 5471, 2016.

[46] Missions | Europa Clipper. https://www.jpl.nasa.gov/missions/europa-clipper/. Accessed Jan. 6, 2021.

[47] SysML Open Source Project - What Is SysML? Who Created It? https://sysml.org/. Accessed Apr. 27, 2021.

[48] SysML Plugin. https://www.nomagic.com/product-addons/magicdraw-addons/sysml-plugin. Accessed Jan. 6, 2021.

[49] ModelCenter Integrate | Model Based Engineering Software | Phoenix Integration. https://www.phoenix-int.com/product/modelcenter-integrate/. Accessed Jan. 13, 2021.

[50] Kim, H., Fried, D., and Menegay, P. Connecting SysML Models with Engineering Analyses to Support Multidisciplinary System Development. In *12th AIAA Aviation Technology, Integration, and Operations (ATIO) Conference and 14th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, Indiana, USA, AIAA2012-5632, p. 5632. 2012.

[51] Ciampa, P. D., and Nagel, B. The AGILE Paradigm: The next Generation of Collaborative MDO. In *18th AIAA/ISSMO multidisciplinary analysis and optimization conference*, Denver, Colorado, AIAA 2017-4137, 2017.

[52] Moerland, E., Ciampa, P. D., Zur, S., Baalbergen, E., Noskov, N., D'Ippolito, R., and Lombardi, R. "Collaborative Architecture Supporting the next Generation of MDAO within the AGILE Paradigm." *Progress in Aerospace Sciences*, Vol. 119, 2020, p. 100637. https://doi.org/10.1016/j.paerosci.2020.100637.

[53] van Gent, I., Aigner, B., Beijer, B., Jepsen, J., and la Rocca, G. "Knowledge Architecture Supporting the next Generation of MDO in the AGILE Paradigm." *Progress in Aerospace Sciences*, Vol. 119, 2020, p. 100642. https://doi.org/10.1016/j.paerosci.2020.100642.

[54] Ciampa, P. D., and Nagel, B. AGILE the next Generation of Collaborative MDO: Achievements and Open Challenges. In *2018 Multidisciplinary Analysis and Optimization Conference*, AIAA 2018-3249, Atlanta, USA, 2018.

[55] Moerland, E., Deinert, S., Daoud, F., Dornwald, J., and Nagel, B. Collaborative Aircraft Design Using an Integrated and Distributed Multidisciplinary Product Development Process. In *30th Congress of the international council for aeronautical sciences*, Deajeon, Korea, 2016.

[56] Home - KE-Chain. https://ke-chain.com/?lang=en. Accessed Jan. 11, 2021.

[57] van Gent, I., and la Rocca, G. "Formulation and Integration of MDAO Systems for Collaborative Design: A Graph-Based Methodological Approach." *Aerospace Science and Technology*, Vol. 90, 2019, pp. 410–433. https://doi.org/10.1016/j.ast.2019.04.039.

[58] Aigner, B., van Gent, I., la Rocca, G., Stumpf, E., and Veldhuis, L. L. M. "Graph-Based Algorithms and Data-Driven Documents for Formulation and Visualization of Large MDO Systems." *CEAS Aeronautical Journal*, Vol. 9, No. 4, 2018, pp. 695–709. https://doi.org/10.1007/s13272-018-0312-5.

[59] Optimus | Noesis Solutions | Noesis Solutions. https://www.noesissolutions.com/our-products/optimus. Accessed Jan. 12, 2021.

[60] RCE. https://rcenvironment.de/. Accessed Jan. 10, 2021.

[61] Boden, B., Flink, J., Mischke, R., Schaffert, K., Weinert, A., Wohlan, A., Ilic, C., Wunderlich, T., Liersch, C. M., Goertz, S., Ciampa, P. D., and Moerland, E. Distributed Multidisciplinary Optimization and Collaborative Process Development Using RCE. In *AIAA Aviation 2019 Forum*, Dallas, Texas, AIAA 2019-2989, 2019.

[62] Boden, B., Flink, J., Först, N., Mischke, R., Schaffert, K., Weinert, A., Wohlan, A., and Schreiber, A. "RCE: An Integration Environment for Engineering and Science.", *arXiv preprint arXiv:1908.03461*, 2019.

[63] Baalbergen, E., Kos, J., Louriou, C., Campguilhem, C., and Barron, J. "Streamlining Cross-Organisation Product Design in Aeronautics." *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, Vol. 231, No. 12, 2017, pp. 2192–2202. https://doi.org/10.1177/0954410017716480.

[64] Liersch, C. M., and Hepperle, M. "A Distributed Toolbox for Multidisciplinary Preliminary Aircraft Design." *CEAS Aeronautical Journal*, Vol. 2, No. 1–4, 2011, pp. 57–68. https://doi.org/10.1007/s13272-011-0024-6.

[65] Liersch, C. M., and Hepperle, M. A Unified Approach for Multidisciplinary Preliminary Aircraft Design. In *CEAS 2009 European Air and Space Conference*, Manchester, UK, 2009.

[66] van Gent, I., la Rocca, G., and Hoogreef, M. F. M. "CMDOWS: A Proposed New Standard to Store and Exchange MDO Systems." *CEAS Aeronautical Journal*, Vol. 9, No. 4, 2018, pp. 607–627. https://doi.org/10.1007/s13272-018-0307-2.

[67] Cloud Computing Services | Microsoft Azure. https://azure.microsoft.com/en-gb/. Accessed Apr. 27, 2021.

[68] Mura, G. L., Riaz, A., Guenov, M., Molina-Cristóbal, A., Chen, X., Sharma, S., Nicholls, K., and Lynas, C. Early Stage Design of High-Lift Devices with System and Manufacturing Constraints. In *AIAA Scitech 2019 Forum*, San Diego, California, AIAA 2019-1035, 2019.

[69] Kulfan, B. M., and Bussoletti, J. E. "Fundamental" Parametric Geometry Representations for Aircraft Component Shapes. In *11th AIAA/ISSMO multidisciplinary analysis and optimization conference*, Portsmouth, Virginia, AIAA 2006-6948, 2006

[70] Guenov, D. M., Molina-Cristóbal, A., Riaz, A., Sharma, S., Murton, A., and Crockford, J. Aircraft Systems Architecting: Logical-Computational Domains Interface. In *31st Congress of the International Council of the Aeronautical Sciences*, Belo Horizonte, Brazil, 2018.

[71] Gondhalekar, A. C., Guenov M. D., Wenzel H., Nunez M., and Balachandran L. K. Neutral Description and Exchange of Design Computational Workflows. In *Proceedings of the 18th International Conference on Engineering Design (ICED 11)*, Lyngby/Copenhagen, Denmark, Vol 1, pp. 113-121, 2011.

[72] Balachandran, L. K., and Guenov, M. D. Object Oriented Framework for Computational Workflows in Air-Vehicle Conceptual Design. In *9th AIAA Aviation Technology, Integration, and Operations Conference (ATIO) and Aircraft Noise and Emissions Reduction Symposium (ANERS)*, Hilton Head, South Carolina, AIAA 2009-7117, 2009.

[73] SQLAlchemy - The Database Toolkit for Python. https://www.sqlalchemy.org/. Accessed Jan. 27, 2021.

[74] APROCONE - Integrated Product Design in the Aerospace Industry. http://reports.raeng.org.uk/datasharing/case-study-6-aprocone/. Accessed Jan. 27, 2021.

[75] Rouvreau, S., Mangeant, F., and Arbez, P. TOICA Innovations in Aircraft Architecture Selection, Uncertainty Management in Collaborative Trade-Off. In *6th IC-EpsMsO - 6th International Conference on Experiments/Process/System Modeling/Simulation/Optimization, The Learning Foundation in Mechatronics (LFME)*, Athens, Greece, 2015.

[76] Final Report Summary - CRESCENDO (Collaborative & Robust Engineering Using Simulation Capability Enabling Next Design Optimisation) | Report Summary | CRESCENDO | FP7 | CORDIS | European Commission. https://cordis.europa.eu/project/id/234344/reporting. Accessed Jan. 9, 2021.

[77] Balachandran, L. K., and Guenov, M. D. "Computational Workflow Management for Conceptual Design of Complex Systems." *Journal of Aircraft*, Vol. 47, No. 2, 2010, pp. 699–703. https://doi.org/10.2514/1.43473.

[78] Nunez, M., Datta, V. C., Molina-Cristobal, A., Guenov, M. D., and Riaz, A. "Enabling Exploration in the Conceptual Design and Optimisation of Complex Systems." *Journal of Engineering Design*, Vol. 23, No. 10–11, 2012, pp. 852–875. https://doi.org/10.1080/09544828.2012.706800.

[79] Guenov, M. D., Riaz, A., Bile, Y. H., Molina-Cristobal, A., and Heerden, A. S. J. "Computational Framework for Interactive Architecting of Complex Systems." *Systems Engineering*, Vol. 23, No. 3, 2020, pp. 350–365. https://doi.org/10.1002/sys.21531.

[80] Guenov, M. D., Nunez, M., Molina-Cristóbal, A., Datta, V., and Riaz, A. "Aircadia – an Interactive Tool for the Composition and Exploration of Aircraft Computational Studies At Early Design Stage." *29th Congress of the International Council of the Aeronautical Sciences*, 2014.

[81] Guenov, M. D., Nunez, M., Molina-Cristóbal, A., Sripawadkul, V., Datta, V. C., and Riaz, A. Composition, Management, and Exploration of Computational Studies at Early Design Stage. In *Computational Intelligence in Aerospace Sciences, Progress in Astronautics and Aeronautics* (M. Vasile and V. M. Becerra, eds.), American Institute of Aeronautics and Astronautics, Reston, 2014, pp. 415–460.

[82] Webix: JavaScript UI Library; HTML5 JS Framework, 102 UI Web Widgets. https://webix.com/. Accessed Apr.

27, 2021.
[83] D3.Js - Data-Driven Documents. https://d3js.org/. Accessed Apr. 27, 2021.
[84] Three.Js – JavaScript 3D Library. https://threejs.org/. Accessed Apr. 27, 2021.
[85] Chen, X., Riaz, A., Guenov, M. D., and Molina-Cristóbal, A. A Set-Based Approach for Coordination of Multi-Level Collaborative Design Studies. In *AIAA Scitech 2020 Forum*, Orlando, FL, AIAA 2020-0320, 2020.
[86] McCullers, L. A. *Flight Optimization System*. In *Proceedings of Recent Experiences in Multidisciplinary Analysis and Optimization*, Hampton, Virginia, pp.24-26, 1984.
[87] Claus, R. W., Evans, A. L., Lylte, J. K., and Nichols, L. D. "Numerical Propulsion System Simulation." *Computing Systems in Engineering*, Vol. 2, No. 4, 1991, pp. 357–364. https://doi.org/10.1016/0956-0521(91)90003-N.