APPLICATION OF AGILE MODEL-BASED SYSTEMS ENGINEERING IN AIRCRAFT CONCEPTUAL DESIGN

Gustavo P. Krupa
Federal University of Itajubá

Keywords: *Model-based systems engineering, agile, systems architecting, aircraft design*

Abstract

One of the challenges of modern engineering design is the amount of data that designers must keep track while performing system analysis and synthesis. This is particularly important in the design process of complex systems such as novel aerospace systems where Modeling and Simulation play an essential role. The Agile Philosophy stems from the field of Software Engineering and describes an approach to development in which requirements and solutions gradually develop through collaboration between self-organizing cross-functional teams and end users. Agile Model-Based System Engineering (AMBSE) is the application of the Agile Philosophy to Model-Based System Engineering. In this paper, AMBSE is accomplished through the application of the Object-Oriented System Engineering Method (OOSEM). OOSEM employs a top-down scenario-driven process that adopts System Modeling Language (SysML) and leverages the object-oriented paradigm to support the analysis, specification, design, and verification of systems. AMBSE assisted by mathematical modeling and safety assessment techniques, is applied to the first design iterations of the main aircraft systems, allowing a comprehensive design exploration. The flight control system was chosen to illustrate the procedure in detail, emphasizing the synthesis of a six degrees-of-freedom model augmented by dynamic inversion control for a hypothetical supersonic transport aircraft satisfying class II MIL-F-8785C handling qualities. It is concluded that Agile Model-Based Engineering improves the early aircraft design process, enabling a smoothly transition from conceptual to preliminary design and system integration.

1 INTRODUCTION

Since the past century, the aerospace industry has been developing and refining the conventional "tube-and-wing" airplane configuration, which is reaching the point of incremental, diminishing returns. As a result, there is a renewed interest in novel/unconventional aircraft systems that must be designed in a cost effective and timely manner, subjected to stringent regulations. Yet, new aircraft systems must be designed to operate in a complex, evolving and broadly-defined world [1], in which requirements and capabilities often change during the aircraft life cycle. It is widely recognized that over 70% of the design features that drive life cycle cost are selected during conceptual design [2]. Under these circumstances, it is generally difficult to define adequate requirements that capture desired system performance and functionality without constraining the design into a sub-optimal design space. Systems Integration is widely accepted as the basis for improving the overall design, the efficiency and the performance of many engineering systems [3]. By adopting a unified mathematical modelling framework that allows efficient performance calculations throughout the system hierarchy, it is possible to bring typically preliminary design activities to the conceptual phase, allowing a much more comprehensive design exploration. This shifts the philosophy of engineering design enabling a systematic development from an integrated system concept to an integrated system product. Ideally, this method should be supported by an underling development philosophy that provides complete coverage of system functionality and requirements tracing. Also, an accurate mathematical description of a system provides the design engineer with the flexibility to perform trade studies quickly and accurately, improving the early design process. Thus, continuous change-friendly holistic approaches supported by mathematical modelling are desirable instead of specifying plain design requirements in the pre-design phase, as usually practiced in the 20th century.

The Agile Philosophy originates from the field of Software Engineering and describes an approach to development in which requirements and solutions gradually develop through collaboration between self-organizing cross-functional teams and end users. This philosophy prescribes adaptive planning, early delivery, incremental changes and continuous improvement, while endorsing rapid and flexible response to change. In a sense, it resembles the ethos of the Skunk Works approach. The original form of the Agile philosophy is given in the Agile Manifesto, a public declaration of intent released by the Agile Alliance [4]. The Agile approach is given by sixteen guiding principles derived from four fundamental statements:

Individuals and interactions *over processes and tools*
Working software *over comprehensive documentation*
Customer collaboration *over contract negotiation*
Responding to change *over following a plan*

The manifesto states an emphasis, not an exclusion of required deliverables, i.e., the items on the left are valued more than the items on the right. It is different from other development philosophies because of its emphasis on the concepts of incremental work, dynamic planning, active project risk reduction, constant validation, continuous integration and frequent verification. It is interesting to note that the Agile mindset has some parallels with the legendary Skunk works approach, as alluded above. In Johnson's [5] view the success of Skunk works was the consistent program management approach and culture, which emphasizes the ability to make immediate decisions and put them into rapid effect, by delegating strong authority to the manager and by employing a small team of strong generalists with system-level thinking. In his book, Johnson [5] gives the early definition of the Skunk works approach:

> *The Skunk Works is a concentration of a few good people solving problems far in advance - and at a fraction of the cost - of other groups in the aircraft industry by applying the simplest, most straightforward methods possible to develop and produce new projects. All it is really is the application of common sense to some pretty tough problems.*

Johnson developed revolutionary aircraft as the P-80, F-104, U-2, C-130, and SR-71, using "14 Rules & Practices [6]" that defines the Skunk Works approach, as described in his autobiography [5]. This management approach fosters creativity and innovation, and has enabled prototyping and development of highly complex aircraft in relatively short time spans and at relatively low cost. The emphasis on Individuals and interactions is denoted by points 2 and 3 of his rules, addressing the need to "the use of strong, but small project offices with 10% to 25% the size of the so-called normal systems". Point 5 calls for a minimum number of reports required and points 8-9 calls for early and continuous verification (working software[1] over comprehensive documentation). Points 7, 10, 12 state the necessity of mutual trust, close cooperation and liaison on a day-to-day basis between involved parties, which parallels the emphasis on customer collaboration by the Agile approach. While, points 1 to 4 and 14 refer to the adoption of a flat organization, where lines of communication are short, making the responsiveness to change rapidly. Point 11 allude to practical earned value management and,

[1] *mutatis mutandis*, verifiable executable models, as explained in section 2.1

lastly but not least, responding to change is alluded by point 6.

2 AGILE MODEL-BASED SYSTEMS ENGINEERING

The methodology used in this work relies on several disciplines and concepts namely: the Agile Philosophy, Model-based system engineering (MBSE, the object-oriented system engineering method (OOSEM) and the System Modelling Language (SysML). The method in MBSE is what defines how the language (SysML) will be used in the context of MBSE. In order to avoid unnecessary confusion it is important to note that:

> **Agile is a development philosophy; MBSE is a paradigm; OOSEM is a method and SysML is a modelling language**.

2.1 Agile Systems Engineering

Systems Engineering is an interdisciplinary activity that focuses more on system properties than on specific technologies and has the overall goal of producing optimized systems to meet potentially complex needs. Although, there are many ways in which to define systems engineering, the following suffices. Systems Engineering is an interdisciplinary activity that focuses more on system properties than on specific technologies and has the overall goal of producing optimized systems to meet potentially complex needs. Although, there are many ways in which to define systems engineering, the following suffices.

> Systems engineering is an iterative process of top-down synthesis, development, and operation of a real-world system that satisfies, in a near optimal manner, the full range of requirements for the system. (Eisner, 2008 [7]; INCOSE, 2015 [8])

Systems engineering has a broad, more holistic view than what may be called "specialty engineering[7]", that usually focus more on the specific development of a particular component or item or is involved with a specific discipline, for instance, aerodynamics or structures. The System engineering process is used when it is necessary to define and allocate requirements to specific engineering disciplines, and in general it should specify the design or technologies only at a high level. Hence, one of the responsibilities of a system engineer is to define systems architecture by performing trade studies to evaluate alternative system architecture, in which system requirements are detailed from a black box perspective. In the aeronautical context, trade-studies should be quantified in terms of figures of merit (FoM) or measures of effectiveness (MOEs).

Agile systems engineering has two main goals. The first is to improve the process of developing specifications that can provide technical orientation to specialty engineering in order to develop systems that satisfies requirements and meets customer's needs. The second main goal of an agile project is to enable follow-on systems development [9]. In agile methods, the system is constructed incrementally and at the end of each iteration, the developing system is ready to be verified for some requirements [9]. Thus, a validation and/or verification process is applied at the end of each iteration to guarantee that the evolving system meets the requirements. To support the statements of the manifesto, the Agile Alliance give a set of 12 principles. In his book, Douglass [9] restates the 12 agile principles [4] for systems engineering. Here only some principles are presented:

Principle 1 *Our highest priority is to satisfy the customer through early and continuous delivery of specifications and systems that demonstrably meet their needs.*
Principle 2 *Welcome changing requirements even late in development. Agile processes harness change for the customer's competitive advantage.*
Principle 4 *Business people and systems engineers must work together daily throughout the project.*
Principle 6 *The most efficient and effective method of conveying information to and within a development team is face-to-face conversation or work products that execute (or simulate).*
Principle 11 *The best architectures, requirements and designs emerge from self-organizing teams.*

Principle 12 *At the regular intervals, the team reflects on how to become more effective, then tunes and adjust its behavior accordingly.*

The incremental, spiral life-cycle developed in this work is shown in figure 1. The project initiates by the capturing of stakeholder requirements and by decomposing the expected functionality. Following the functional analysis, a requirement analysis is realized. Then, it is possible to define the logical architecture which is done by establishing a hierarchy within the system and by allocating requirements to its corresponding functions. Based on the logical architecture, initial architectures are proposed. These are analyzed and new candidate architectures are synthesized in order to fit the desired functionality and its requirements. The proposed architecture is validated accordingly and the overall system is verified. The cycle proceeds in spiral meaning that each subsystem and its components, if necessary, are developed in the same incremental way. The process is iterative and at each cycle requirements, functionality and system elements are continuous updated to reflect the new available information. This incremental development is essentially the same that one typically follows when designing by first principles.

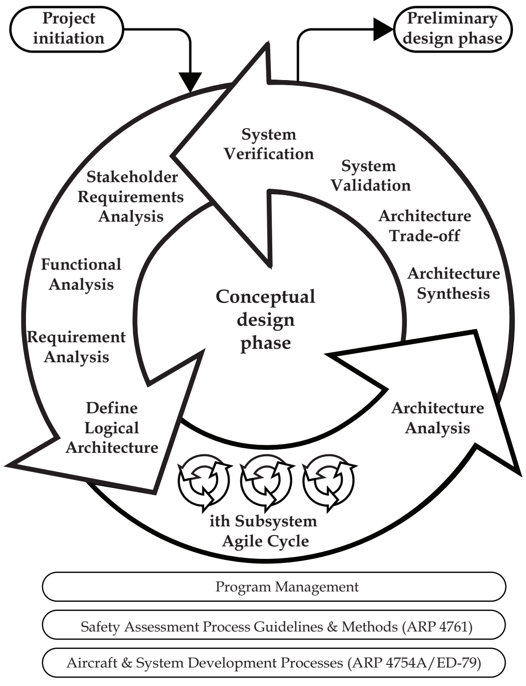

Fig. 1 Agile System Engineering process for aircraft conceptual design

2.2 Model-Based Systems Engineering (MBSE)

Blanchard [10] defines Model Based Systems Engineering (MBSE) as the formalized application of graphical modelling to support system engineering activities beginning in the conceptual design and continuing throughout the entire life cycle. MBSE supports and enhances the ability to conduct system engineering tasks such as requirements capturing, design, analysis, validation and verification. MBSE is often contrasted with a traditional textual-based approach to Systems Engineering. In MBSE, the primary artifact of the system engineering process is the system model. On the other hand, in a textual-based system engineering approach, there is often considerable information generated about the system contained in textual documents. This information is often difficult to maintain and to assess in terms of its quality. In a MBSE approach, much of this information is captured in a system model.

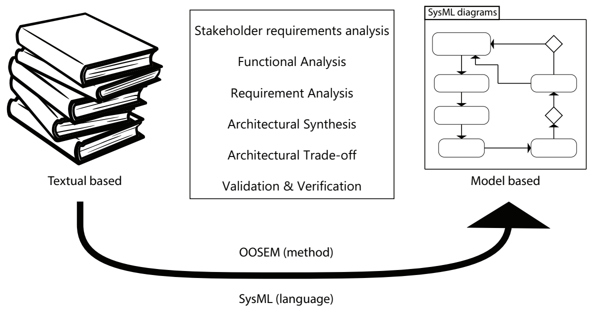

Fig. 2 Transition from SE to MBSE

2.3 Object oriented systems engineering method (OOSEM)

The Object-oriented Systems Engineering Method (OOSEM) is a MBSE approach that leverages object-oriented concepts and adopts SysML to facilitate the capture and analysis of requirements and design information to specify complex systems [11]. OOSEM is usually applied recursively and interactively at each level of the system hierarchy and it employs a top-down approach to specifying, analyzing, verifying, design and development. In agile methods, the system is incrementally constructed at each iteration and OOSEM provides the flexibility to accommodate changing requirements and design evolution, making it an ideal method for the purposes of this work, as long as it is tailored for agile, as the INCOSE handbook [8] remarks.

2.4 Systems Modelling Language (SysML)

The Systems Modelling Language (SysML) from the Object Management Group (OMG) has emerged from the Unified Modeling Language [12] (UML), the de facto standard for modelling in software engineering. Friedenthal [11] defines SysML as a general-purpose modelling language for systems engineering applications that supports the specification, analysis, design, validation and verification of systems and systems-of-systems. SysML models systems aspects that may be classified into three groups: the behavioral aspect, its structure and its requirements.The reader is referred to Friedenthal [11] and the SysML formal specifications [13] for a comprehensive reference on SysML.

3 SYSTEMS DEVELOPMENT WITH AMBSE

The last section presents AMBSE in general terms, not particularly focusing on its application in aircraft design and development. This section shows that not only AMBSE can enhance the conceptual design process, but is also consistent with ARP-4754A and ARP-4761 guidelines.

3.1 Overview of ARP-4754A

The document ARP4754A [14] provides guidelines for the development cycle of aircraft systems, for the planning of the development process and guidelines for showing compliances with regulations. In general, aircraft systems show a high degree of interaction between systems. Thus, they have many modes of failure that affect the safety of the aircraft. ARP 4754A provides a methodology to mitigate development errors and provide a guideline for the assigning the adequate assurance level that errors in the development cycle have been identified and corrected. This is realized by assigning a Functional Development Assurance Level (FDAL) to the top-level Function, based on its most severe Top-Level Failure Condition Classification in accordance with Table 1.

Table 1 Top-Level Function FDAL assignment

Severity	FDAL Assignment
Catastrophic	A
Hazardous/Severe Major	B
Major	C
Minor	D
No Safety	E

AMBSE benefits from the use of ARP 4754A and ARP 4761 practices, since these technical standards imposes design discipline and development structure, ensuring that both operational and safety requirements are fully realized and substantiated. Also, AMBSE facilitates the use of these practices by providing a design environment where requirements and solutions evolve by small verifiable increments through collaboration between self-organizing cross-functional teams and end users.

4 APPLICATION OF AMBSE IN CONCEPTUAL DESIGN

This section presents the application of AMBSE to the first design iterations during the conceptual design phase. AMBSE is applied to both the aircraft, system, and component level following the process described in figure 1. The component level is used to illustrate how AMBSE may support the so-called Post-Tier 1 [15] supply chain trend, which involves more vertical integration and restructured responsibilities between OEM and its suppliers. AMBSE supports an approach in which the aircraft designer is aware of the interactions and implications between systems and different disciplines. This is especially important in conceptual design, which is an intense exploratory phase due to its iterative process structure, involving feedback loops and successive refinements. In addition, the emphasis on the practice of first principles design within AMBSE results in rapid design-responses, allowing requirements and solutions to gradually develop through collaboration between cross-functional teams and end users, as prescribed by the Agile Philosophy.

4.1 Design Methods & Tools

The AMBSE process described in section 2.1 and shown in figure 1 is represented as a SysML activity diagram in figure 3. The activity box inside the diagram describe the system engineering activities. After the specific system/subsystem architecture, it is compared with alternative proposals. Once a candidate architecture is considered ready, it is passed to Integrated Product Development Teams (IPDTs) for preliminary design. The standards ARP-4754A and ARP-4761 are used to requirements generation during the entire process.

AMBSE requires an integrated development environment (IDE). In this work, Eclipse [16] Papyrus

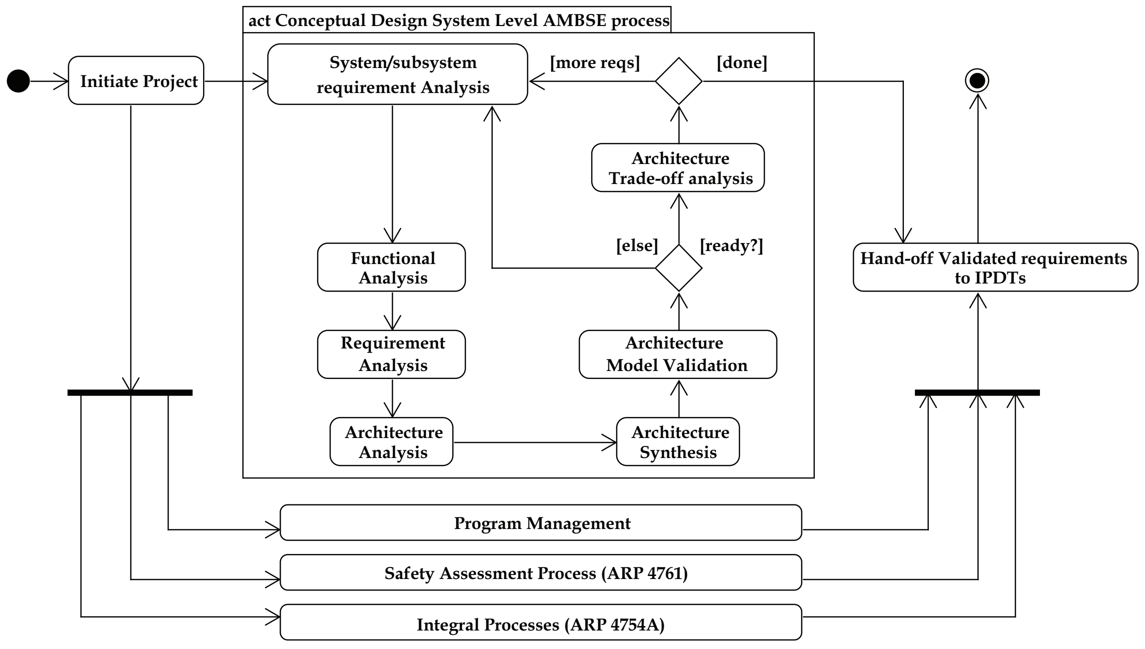
act Conceptual Design System Level AMBSE process
Initiate Project
System/subsystem requirement Analysis
[more reqs]
[done]
Architecture Trade-off analysis
[else]
[ready?]
Functional Analysis
Requirement Analysis
Architecture Analysis
Architecture Synthesis
Architecture Model Validation
Hand-off Validated requirements to IPDTs
Program Management
Safety Assessment Process (ARP 4761)
Integral Processes (ARP 4754A)

Fig. 3 AMBSE Delivery Process During Conceptual Design Phase

[17] was chosen because it is open-source, offers UML/SysML modelling capabilities and it contains an extensible plug-in system for customizing the environment. In particular, Papyrus was augmented with Massif [18] and Epsilon [19] plug-ins, allowing to transfer SysML activity diagrams to subsystem blocks in the Simulink environment. Engineering analysis necessitate the creation of dynamic models. The use of the object-oriented paradigm with bond graph modelling facilitates the process of deriving the necessary differential equations. The software 20-sim [20] was used in this work to model systems with the bond graph formalism and also to generate the respective s-functions, which can promptly be used in a Simulink model with great numerical computational performance.

Microsoft© Excel™ is used as a technical calculation and report creation tool, since it is nearly universally used across the world and because it negative aspects can be managed. All of the spreadsheets, conforming to the same format and layout. The traditional design approach as in Roskam [21], Nicolai [22], Datcom [23], Kuchemann [24], Dubbel [25] and Hoerner [26, 27] was implemented in a spreadsheet customized with XL-Viking[28] plug-in. Most of the spreadsheets use the XL-Viking Add-in to display and audit the calculations. The XL-Viking add-in contains easy to use functions that show all the numbers or variables in each Excel formula. This ensures accuracy and traceability of the calculations, allowing significant and valuable time is saved in checking and auditing of calculations while keeping all of the calculations live and writing and updating reports is much quicker and easier. This feature also facilitate the use of conceptual development and design-related analytical tools (such as risk assessment matrix and sensitivity analysis), described in Goldberg et all [29]. This spreadsheet approach allows Python scripting through XL-Wings [30] plug-in to facilitate the data transferring to the Simulink environment and to generate input to the SUAVE conceptual level aircraft design environment. The design environment SUAVE [31] was used to both aerodynamic and performance analyses. The plug-in gendoc [32] is used in the Eclipse environment to generate word files containing the corresponding diagrams. Recurrent engineering calculations were also inserted in the generated documents.

4.2 System Requirements Analysis

The AMBSE approach was applied in the design of a hypothetical airplane designated as Kr-206 which is intended to provide ways to test and develop advanced design techniques and methodologies. In particular, SST aircraft design demands special attention to the flight control system, where many factors differ significantly from subsonic aircraft. Most of the stakeholder requirements were inferred from technical literature [33, 34, 35], except the range requirements. The regulation FAA Part 25 [36] does not establishes flying qualities criteria. However, the document flight control design best practices [37] recommends that the MIL-F-8785C [38] be followed as a guide for flying qualities criteria. The table 2 shows a partial list of the requirements gathered from both Part 25 and MIL-F-8785C.

4.3 Functional Analysis

The purpose of this activity in the systems engineering process is to iteratively identity the functions that the system must perform. Functional analysis is essential to the application of ARP-4754A/ARP-4761 standards, because of that in the methodology followed in this work it is given a specific phase in the process. This activity establishes basic aircraft level performance, and operational requirements, which is accomplished by arranging the functions into logical sequences, decomposing top-level functions into lower-level functions, and allocating performance requirements generated from the higher-level functions in the hierarchy to the lower-level ones. The output of this process is the functional architecture of the system, that is, a description of the system, in terms of its functionality. According to ARP-4754A [14], the output of this activity is a list of aircraft level functions and associated function requirements and interfaces for these functions.

4.4 Logical Architecture Definition

The logical architecture establishes the structure and boundaries within which specific system design are implemented to meet all of the established safety and technical requirements. In practice, this the logical architecture definition phase, requirement and analysis functional and the allocation of requirements are tightly-coupled iteratively processes. Since functional and performance requirements originate at the highest levels of the system hierarchy, these activities must be continuously repeated to define the logical architecture at ever greater levels of detail. This process generates many types of requirements which include: independence, probabilistic, qualitative, availability, integrity, monitoring, operational and maintenance requirements and in latter iterations Function Development Assurance Level (FDAL). At aircraft level the safety requirements are generated from the aircraft FHA based on top-level aircraft functions previously defined. At system level, the safety requirements are all those system level requirements generated from the system FHA which are decompositions of the aircraft level safety requirements. At the next level down the requirements are all those aspects of the system which allow the safety objectives associated with the system FHA classifications to be satisfied. The output of this process generates many elements which need to be organized via SysML packages. In addition to that, the preset work adopted the Joint Aircraft System/Component (JASC [39]) code tables. The JASC numbering provides a consistent framework for the aircraft technical documents. At the item (component and subcomponent) level, S1000D [40] were used.

4.5 Architecture Synthesis and Analysis

The activity architecture synthesis and analysis comprises the allocation of functionality and corresponding different kinds of requirements to high-level physical elements. Ideally, more than one candidate system architecture should be considered in this activity. These candidate architectures are then iteratively evaluated using functional and performance analysis. In latter iterations, when the candidate architecture contains sufficient detail, it is possible to to apply preliminary phase ARP-4761 [41] Preliminary Aircraft Safety Assessment (PASA)/Preliminary System Safety Assessment (PSSA) processes to establish the feasibility in meeting aircraft and functionality and top level safety requirements assigned to the system.

Aircraft Level The hypothetical aircraft development in this work features a double cranked-arrow wing. A feature of this wing is that its aerodynamic center shifts as the Mach number increases, moving towards the tail cone of the aircraft [42]. A major penalty of this type of wing is that the drag increases due to the required deflection of the control surfaces needed to compensate an aircraft, in transonic regime. In supersonic regime, this penalty is even greater and the stability of the phugoid mode is often reduced. In such cases, it is desired to move the C.G during flight, which can be accomplished by integrating the fuel system with the flight control system. By transferring fuel, it is possible to maintain the static margin during transonic and supersonic regime to only 3% of the mean aerodynamic chord. This feature highlights the benefit of early systems integration in conceptual design that AMBSE makes possible and manageable. The aircraft level AMBSE is discussed in section 4.2 to 4.4.

Table 2 Partial list of longitudinal flying qualities requirements

Identification	Name Type	Specification
R-ST-02	Longitudinal static stability	There shall be no tendency for airspeed to diverge aperiodically when the airplane is disturbed from trim with the cockpit controls fixed and with them free.
R-FCS-FQ-04-02-01-B	Short-period damping	The equivalent short-period damping ratio, ζ_{SP}, shall be within the limits $0.30 < \zeta_{SP} < 2.0$.
R-FQ-05-03	Pilot-induced oscillations	There shall be no tendency for sustained or uncontrollable oscillations resulting from the efforts of the pilot to control the airplane.

System Level In the conceptual design, it was decided that the aircraft shall not have a horizontal stabilizer in order to improve its aerodynamic performance, thus it requires a fly-by-wire system to augment its longitudinal stability characteristics. in figure 4, an initial flight control system architecture is proposed, in which all primary flight control surfaces are all electrically-controlled and hydraulic activated. The FCS interfaces with hydraulic and electrical systems. The actuators are powered by the aircraft blue, green and yellow hydraulic lines. The flight control surfaces are powered by a combination of hydraulic and electro-hydrostatic actuators. It is assumed that the FCS possess by three primary computers and two secondary computers that process pilots and autopilots inputs according to normal, alternate or direct flight control laws, as shown in figure 4. The nose up and down movement are controlled by 4 elevons.

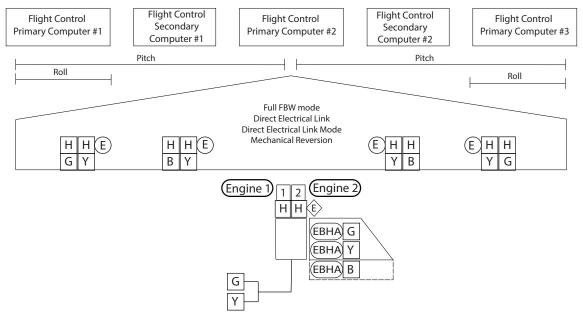
Flight Control Primary Computer #1
Flight Control Secondary Computer #1
Flight Control Primary Computer #2
Flight Control Secondary Computer #2
Flight Control Primary Computer #3
Pitch
Roll
Pitch
Roll
Full FBW mode
Direct Electrical Link
Direct Electrical Link Mode
Mechanical Reversion
H H E
G Y
H H E
B Y
E H H
Y B
E H H
Y G
Engine 1
1 2
Engine 2
H H E
EBHA G
EBHA Y
EBHA B
G
Y

Fig. 4 Proposed Initial FCS Architecture

From the simple candidate architecture schematics shown in figure 4, it is possible to consistently derive its corresponding bond graph by substituting each identified system by a word bond graph. A SysML block definition diagram is draw to represent its structure and interfaces with associated functions and requirements. Each element is the proposed architecture is then modeled as a bond graph using the software 20-Sim [20]. This software is capable of exporting the bond graph model and its sub-models as s-functions, which can readily be integrated in the Simulink model.

An initial candidate architecture were proposed for the fuel system, see figure 5. It was designed to provide data for the total fuel volume required, the size, location and number of fuel tanks needed and the number of fuel pumps, its location and the required capacity of fuel pumps and fuel lines. The engine fuel flow was obtained in this stage by multiplying the maximum required thrust by the associated fuel consumption. Although it is reasonable to assume that the number of tanks in order to keep the cost to a minimum and reduce weight, in this work, the size, location and number of tanks were driven by stability requirements in terms of the desired location of C.G for different loading scenarios. The sizing of fuel lines and the determination of necessary fuel pump pressures were calculates using [43]. The simple schematics in figure 5 along with first principle calculations was sufficient to generate a bond graph representation, which was included in the Simulink model as a s-function.

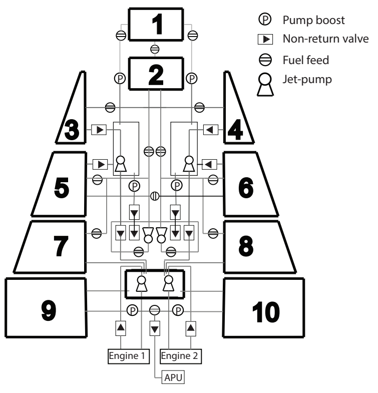

Fig. 5 Proposed Initial Fuel System Architecture

Component Level As shown in figure 4, the initial FCS architecture employs Electro-Hydrostatic Actuator (EHA) and Electro-backup-Hydraulic Actuator (EBHA). The design of an EHA (see figure 6) requires multi-domain modelling capability, since on it mechanical, hydraulic, thermal and electrical domains interact with each other. Therefore, bond graphs are an ideal tool for modelling such components. Following Langlois et all [44], the following EHA components are modeled by the Bond Graph shown in figure 7: Electrical motor, hydraulic pump, accumulator, associated hydraulics (two valves and a bypass valve), hydraulic cylinder, the mechanical actuation (four-bar mechanism) and control surface. it is assumed that the EHA is supplied with a constant DC voltage source. In practice, the EHA is fed with a three-phase AC power that supplies power drive electronics, which in turn, drive a variable speed pump together with a constant displacement hydraulic pump [45].

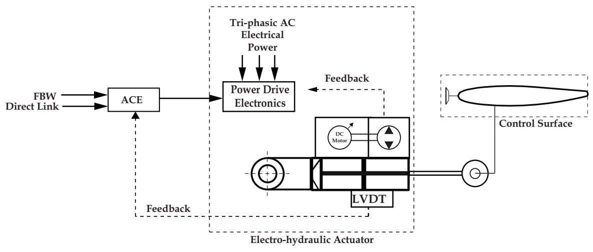

Fig. 6 EHA schematics

Table 3 presents a partial list of servo-actuator specifications deduced from the AMBSE approach. They were generated from the complete bond graph model. It's nominal parameters were calculated using standard mechanical engineering methods that were programmed into design spreadsheets. The complete list has 30 requirements, including performance and functional requirements and it is intended to complement and foster discussion with stakeholders and suppliers.

4.6 Validation and Verification

Validation and Verification comprise independent procedures that are used together for checking that the system meets its intended functions, its requirements and specifications. In the present work, both procedures are realized in MATLAB/Simulink in a six-degree-of-freedom flight dynamics model. A briefly discussion of this model and control laws are contained in the appendix. The following requirements (presented at table 2) were validated and verified: pilot-induced oscillations, short-period damping, short-Period frequency and acceleration sensitivity, thus the system satisfies the short-period Response. The longitudinal response to a elevon step command for the non-augmented aircraft is shown in figure and the longitudinal behavior of the aircraft using the dynamics inversion control law is shown in figure 9.

The system was verified in low subsonic cruise conditions (M = 0.70 and FL340) for two different control laws for: Gain scheduling and dynamic inversion. They were compared using the Control Anticipation Parameters (CAP). The CAP associated with Gain scheduling is 3.435, while dynamic inversion results in a CAP of 2.10. The aircraft conceptual design specification is shown in table 4. A side and a top view of the hypothetical aircraft Kr-206 designed with AMBSE are shown in figure 10.

5 CONCLUSIONS

This paper demonstrated the use of the Agile Philosophy in the aircraft conceptual design. The design of the flight control system is selected to illustrate the procedure in detail and it is concluded that Agile Model-Based Engineering greatly improves the early aircraft design process, allowing a smoothly transition from conceptual to preliminary design. It is shown that verifiable models are required for agile systems engineering to enable design studies across all disciplines and constraints. OOSEM and SysML provide the flexibility to accommodate changing requirements and design evolution, making them good candidates for modelling within the Agile Philosophy. This not only ensures that at the end of each iteration, the maturing system design meets the requirements from early on, but guaranties later a smoothly transition from conceptual to preliminary design. The mathematical models necessary to develop the verifiable models in Simulink can be easily derived with the Bond Graph approach. Moreover, bond graph models used in a objected-oriented way harmonizes with the methodology described and can be used by engineers to perform straightforward numerical analysis in addition to gain qualitative insight, aiding the designer especially in the early stages of design and integration. The common spreadsheet approach enhanced with add-ins is capable of ensuring accuracy

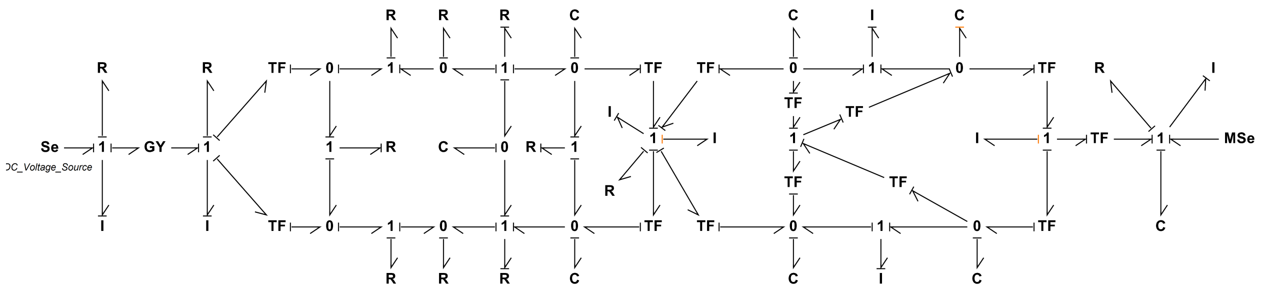

Fig. 7 Bond Graph representation of the EHA

Table 3 Partial list Servo-actuator Specifications

Identification	Name Type	Specification
Req-EHA-F-01	Operating Pressure	The required operating pressure, P_s, is 26 MPa.
Req-EHA-F-10	Maximum operating pressure	The actuator should be designed for a maximum operating pressure, P_{max_A}, of 23.6 MPa.
Req-EHA-F-11	Maximum acting force	The actuator should be designed for a maximum force, F_{max_A}, of 14430 N.
Req-EHA-F-15	Extended Actuator Length	The actuator should be designed for an extended length of 560 mm.

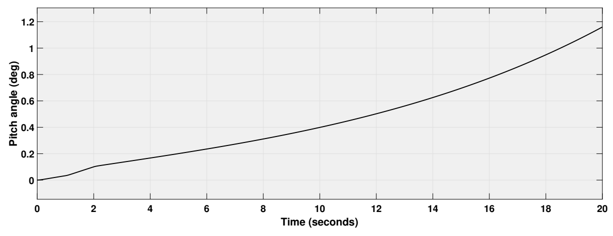

Fig. 8 Non-augmented longitudinal response

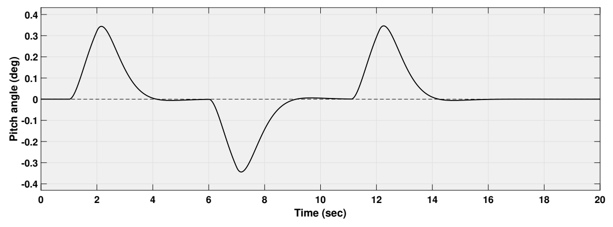

Fig. 9 Augmented longitudinal response

and traceability of the initial parameters calculations. The combination of mathematical modelling, Safety Assessment and system design techniques provides valuable insights in the conceptual design process, especially when applied to lower level aircraft systems. Within the Agile Philosophy, the engineering experience and creativity still remains the essential keys to the successful development of the system.

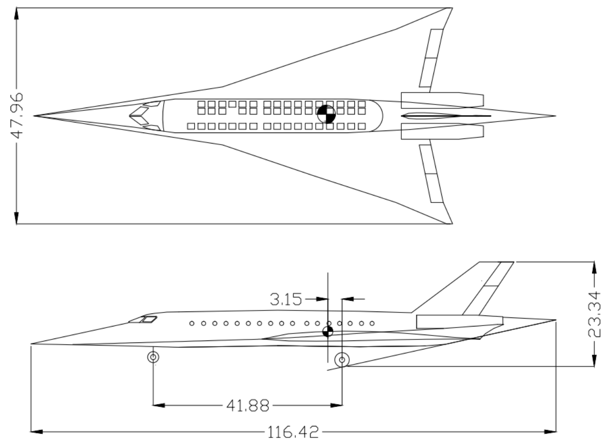

Fig. 10 Hypothetical SST designed using AMBSE

Table 4 Kr-206-A Partial list of specifications

Wing Area	127.18 m^2
Aspect Ratio	1.58
Wing Loading	3112 N/m^2
Number of Passengers	37
Maximum Take-off Weight	40361.5 kg
Fuel Maximum Weight	13935.3 kg (34% MTOW)
Empty Weight	21597.8 kg
Cruise Altitude	10058.4 m (33000 ft) 14325.6 m (47000 ft)
Cruise Mach	M = 0.93 and 1.4

References

[1] German B and Daskilewicz M. *An MDO-Inspired Systems Engineering Perspective for the" Wicked" Problem of Aircraft Conceptual Design*. In: 9th AIAA Aviation Technology, Integration, and Operations Conference (ATIO) and Aircraft Noise and Emissions Reduction Symposium (ANERS). 2009. p. 7115.

[2] Nicolai L M. *Lessons Learned: A Guide to Improved Aircraft Design*. AIAA American Institute of Aeronautics & Ast. (June 15, 2016)

[3] Moir I. and Seabridge A. *Design and development of aircraft systems*, John Wiley & Sons, Hoboken, New Jersey, USA, 2012.

[4] Agile Alliance. *Agile manifesto*. Available at http://www.agilemanifesto.org, v. 6, n. 1, 2001.

[5] Johnson C L. *Kelly: more than my share of it all*. Smithsonian Institution. 2012.

[6] Kelly's 14 Rules & Practices available at https://lockheedmartin.com/us/aeronautics/skunkworks/14rules.html

[7] Eisner H. *Essentials of Project and Systems Engineering Management*, John Wiley & Sons, Inc, 2002.

[8] International Council on Systems Engineering. *Systems Engineering Handbook âĂŞ A Guide for System Life Cycle Processes and Activities*, 2015.

[9] Douglass B P. *AGILE systems engineering*, Morgan Kaufmann, Burlington, Massachusetts, USA, 2015.

[10] Blanchard B S and Fabrycky, W J. *Systems engineering and analysis*, Pearson, USA, 1998.

[11] Friedenthal S., Moore, A., and Steiner, R. *A practical guide to SysML: the systems modeling Language*, Morgan Kaufmann, Burlington, Massachusetts, USA, 2014.

[12] Object Management Group®. *Information technology - Object Management Group Unified Modeling Language (OMG UML)*, version 2.5.1, 2017. Available at https://www.omg.org/spec/UML/2.5.1/PDF.

[13] Object Management Group®. *Information technology - Object Management Group Systems Modeling Language (OMG SysML)*, version 1.4, 2015. Available at https://www.omg.org/spec/SysML/1.4/PDF.

[14] SAE. *ARP-4754A: Aerospace Recommended Practice: Guidelines For Development Of Civil Aircraft and Systems*. December, 2010.

[15] Michaels K. *Post-Tier 1: The next era in aerospace supply chain evolution? Aviation Week* & Space Technology, 18 May 2017.

[16] Eclipse Foundation. *The Eclipse Foundation open source community website*. (2018). https://www.eclipse.org/

[17] Eclipse Foundation. *Papyrus open source community website*. (2018). https://www.eclipse.org/papyrus/

[18] Massif: MATLAB Simulink Integration Framework for Eclipse. https://github.com/viatra/massif

[19] Eclipse Epsilon team. *Epsilon* https://www.eclipse.org/epsilon/

[20] Controllab Products B.V. *20-sim.* , Drienerlolaan 5 HO-8266, 7522 NB Enschede, The Netherlands., www.20sim.com.

[21] Roskam, J. *Airplane design*, DARcorporation, Lawrence, Kansas, USA, 1985.

[22] Nicolai L M and Carichner G. *Fundamentals of aircraft and airship design*. American Institute of Aeronautics and Astronautics, 2001.

[23] Hoak E. and Ellison D. *USAF Stability & Control DATCOM*, AFFDL-TR-79-3032, Flight Control Division, Air Force Flight Dynamics Laboratory, Wright-Patterson Air Force Base, Ohio, USA, 1972.

[24] Kuchemann D. *The aerodynamic design of aircraft. Progress in aeronautical sciences*, Pergamon, London, 1978.

[25] Dubbel H and Davies B J. *Handbook of mechanical engineering*. Springer Science & Business Media, 2013.

[26] Hoerner S F. *Fluid-dynamic drag: practical information on aerodynamic drag and hydrodynamic resistance*. Published by the Author, Bakersfield, California, USA, 1965.

[27] Hoerner S F and Henry V Borst H V. *Fluid-Dynamic Lift, Practical Information on Aerodynamic and Hydrodynamic Lift.* 1975.

[28] XL-Viking. *Excel Mathematics Visualisation Add-in* (c) 2015-2018 Abbott Aerospace SEZC

Ltd and Knut Gjelsvik.

[29] GOLDBERG, B.E., EVERHART, K., STEVENS, R., BABBITT III, N., CLEMENS, P., STOUT, L. *System engineering toolbox for design-oriented engineers*, NASA Reference Publication 1358, 1994.

[30] ZOOMER ANALYTICS LLC.. *XL-Wings: Python for Excel*. V0.11.7, Feb 5, 2018.

[31] MacDonald T, Clarke M, Botero E, Vegh J M, Alonso J J. *"SUAVE: An Open-Source Environment Enabling Multi-fidelity Vehicle Optimization"*, 16th AIAA Multidisciplinary Analysis and Optimization Conference, Denver, CO, June 2017.

[32] Eclipse Gendoc team *Gendoc* `https://www.eclipse.org/gendoc/`

[33] Wolf R. *A summary of recent supersonic vehicle studies at Gulfstream Aerospace*. In: 41st Aerospace Sciences Meeting and Exhibit, 2003.

[34] Henne P. *A Case for small supersonic civil aircraft*. Journal of Aircraft, v.42, n.3, p. 765-774, 2005.

[35] Smith H. *A review of supersonic business jet design issues*. The Aeronautical Journal, v.111, n.1126, p. 761-776, 2007.

[36] FAA. *Systems Engineering Manual*, Version 3.1. Federal Aviation Administration. (2006)

[37] NATO, RTO. *Flight Control Design - Best Practices*, RTO-TR-029, December, 2003.

[38] Moorhouse D., Woodcock, R. *US Military Specification MIL-F-8785C*. US Department of Defense, 1980.

[39] Air Transport Association of America. *Joint Aircraft System/Component Code Table and Definitions*

[40] ASD/AIA. *S1000D, "International specification for technical publications using a common source database"*.

[41] SAE. *ARP-4761: Aerospace Recommended Practice: Guidelines & Methods for Conducting the Safety Assessment on Civil Airborne Systems and Equipment*, December, 1996.

[42] Stengel, R. F. *Altitude stability in supersonic cruising flight*. Journal of Aircraft, 7(5), 464-473, 1970.

[43] Avallone, E. A., Baumeister, I. T., & Sadegh, A. *Marks' Standard Handbook for Mechanical Engineers*. New York, McGraw-Hill, 2006.

[44] Langlois, O. et al. *Bond Graph Modeling of an Electro-Hydrostatic Actuator for Aeronautic Applications*. In: IMACS World Congress. 2005.

[45] Moir I and Seabridge A. *Aircraft systems: mechanical, electrical and avionics subsystems integration*, John Wiley & Sons, Hoboken, New Jersey, USA, 2011.

6 Contact Author Email Address

mailto:gustavokrupa@gmail.com

Copyright Statement