MODEL-BASED SYSTEMS ENGINEERING APPROACH WITH SYSML FOR AN AUTOMATIC FLIGHT CONTROL SYSTEM

Haluk Altay and M. Furkan Solmazgül

Teknopark Istanbul, Turkish Aerospace, Istanbul, Turkey

ABSTRACT

Systems engineering is the most important branch of engineering in interdisciplinary study. Successfully performing a multidisciplinary complex system is one of the most challenging tasks of systems engineering. Multidisciplinary study brings problems such as defining complex systems, ensuring communication between stakeholders, and common language among different design teams. In solving such problems, traditional systems engineering approach cannot provide an efficient solution. In this paper, a model-based systems engineering approach is applied with a case study and the approach is found to be more efficient. In the case study, the design of the helicopter automatic flight control system was realized by applying model-based design processes with integration of tools. Requirement management, system architecture management and model-based systems engineering processes are explained and applied of the case study. Finally, model-based systems engineering approach is proven to be effective compared with the traditional systems engineering methods for complex systems in aviation and defence industries.

KEYWORDS

Model-Based Systems Engineering, Automatic Flight Control System, SysML.

1. INTRODUCTION

In order to understand Model-based systems Engineering, it is necessary to know the definition and scope of the systems engineering. The definition of systems engineering is defined as follows in the references.

“Systems engineering is an interdisciplinary approach and means to enable the realization of successful systems.” [1]

“Systems engineering is a discipline that concentrates on the design and application of the whole (system) as distinct from the parts. It involves looking at a problem in its entirety, considering all the facets and all the variables and relating the social to the technical aspect.” [2]

Considering these reference definitions, within the scope of this case study, the definition of systems engineering is made as follows.

“Systems engineering is a multidisciplinary and common mind approach that ensures successful realization of systems.”

David C. Wyld et al. (Eds): CMC, NCO, SOFT, CDKP, MLT, ICAITA - 2021
pp. 01-19, 2021. CS & IT - CSCP 2021 DOI: 10.5121/csit.2021.111101

The definition of model-based systems engineering is defined as follows in the references.
"Model-based systems engineering (MBSE) is the formalized application of modelling to support system requirements, design, analysis, verification and validation activities beginning in the conceptual design phase and continuing throughout development and later life cycle phases." [3]
Considering these reference definitions, within the scope of this case study, the definition of model-based systems engineering is made as follows.

"Model-based systems engineering is an approach to successfully realize systems driven by a model, with a consistent set of views that reflect multiple perspectives of the system."

The traditional systems engineering definition has been made to the systems engineering activities carried out before the model-based systems engineering approach. Traditional systems engineering has three main problems, such as being inadequate in defining complex systems, communication between stakeholders, and common language and interpretation among different design teams. In the case study, model-based systems engineering processes are applied and as a result, main problems are eliminated. Model-based systems engineering approach has multiple advantages over traditional systems engineering.

- Automatic generation of most of system documents by using the developed models.
- Since the models can be measured, it is easy to control and manage in complex systems.
- Consistency for all information in the system architecture thanks to the models.
- Ensuring traceability in the life cycle stages of the system with using SysML for modelling and maintaining tool integration.
- Easier to access information since a certain systematic is applied while the model is being established.
- More understandable communication establishment thanks to the representation of requirements as a model and the use of a common language for this model.
- Improving communication as a result of the establishment of common terminology and concepts between all stakeholders and design teams of a system.

It is important for companies that model-based systems engineering benefits are directly related to cost, time or resource savings. Adapting the model-based systems engineering approach to reflect the company's working principles is the most critical point for the efficiency of this process.

2. DESIGN METHOD

In the INCOSE System Engineering Handbook document that there are 6 different methods of model-based systems engineering. [1] These methods are INCOSE Object-Oriented Systems Engineering Method (OOSEM), IBM Rational Telelogic Harmony-SE, IBM Rational Unified Process for Systems Engineering (RUP-SE), Vitech MBSE Methodology, JPL State Analysis (SA) and Dori Object – Process Methodology (OPM).

The methods that mentioned above have been examined and a suitable method has been determined for the flight control system. The method created is shown in Figure 1.

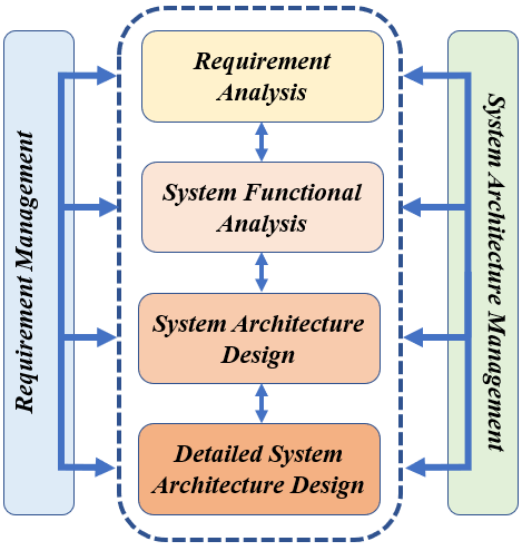


Figure 1. Model-Based Systems Engineering (MBSE) process

Model-based System Engineering processes; requirements analysis, system functional analysis, system architecture design and detailed system architecture design. Requirement and system architecture management are required throughout MBSE processes. By this means, process traceability and an iterative design are provided. The method followed from the customer requirements that are the input of the MBSE process to the system requirements and the system architecture that are the outputs of the process are shown in Figure 2.

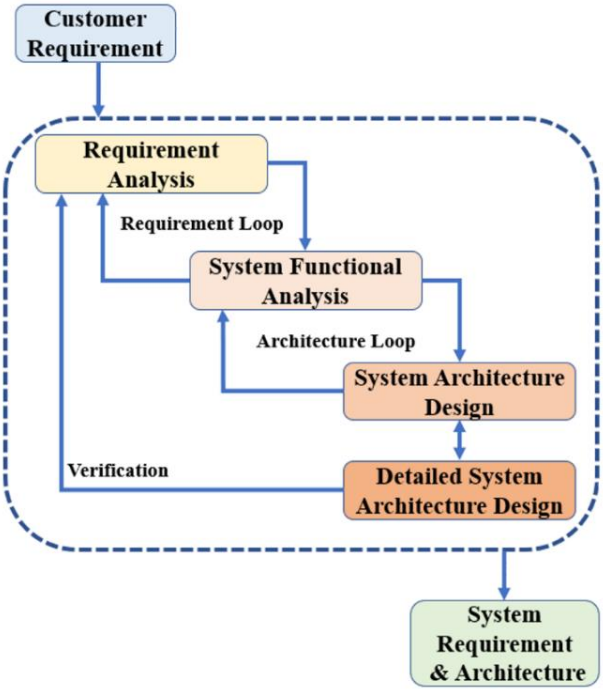


Figure 2. Model-Based Systems Engineering (MBSE) process with input and outputs [4]

In the case study of helicopter automatic flight control system, Model-based System Engineering process was carried out with reference to the ARP4754A document and the relevant sections of the DO-178C / 331 standard document listed below.

- ARP-4754A Section 4.5 Allocation of System Requirements to Items
- ARP-4754A Section 5.3 Requirements Capture
- ARP-4754A Section 5.4 Requirements Validation
- DO-178C/DO-331 Section 2.1 System Requirements Allocation to Software
- DO-178C Table A-2 Software Development Process
- DO-178C Table A-3 Verification of Outputs of Software Requirement Process

3. DESIGN

Helicopter automatic flight control system architecture design was realized by using SysML with the Model-based systems engineering approach. The design process was carried out in accordance with the method described above.

3.1. Requirement Management

Requirement management is an iterative process that continues throughout model-based systems engineering processes of the case study. Requirement management covers the following processes:

- Definition of requirements
- Validation of requirements
- Traceability and verification of requirements
- Transfer and synchronization of requirements.

In the requirement management, managing the requirements with a single software ensures that each stage is carried out more efficiently. In this case study, DOORS software was selected for requirement management processes. The relationship between the requirements in the DOORS and the model elements in the system architecture which is modelled with SysML was created by using IBM Rational Rhapsody.

3.1.1. Definition of Requirements

Within the scope of the case study three set of requirements were created by using DOORS:

- Contractual requirements that defining the behaviours that are targeted to occur in the system and received from costumer.
- System requirements that defining all functions and properties of system.
- Software requirements that taken as reference when designing flight control computer and implementing the software.

Requirement sets are defined in a hierarchy as in Figure 3. Thus, the levels at which the requirements are created, and which standards are used as the source when deriving the requirement sets are shown in Figure 3.

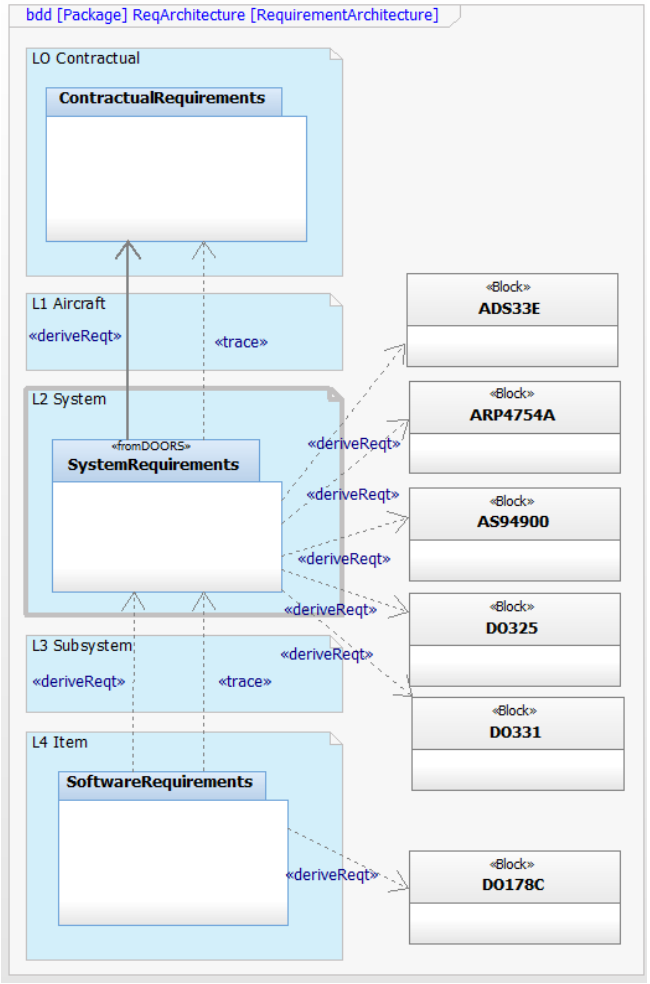
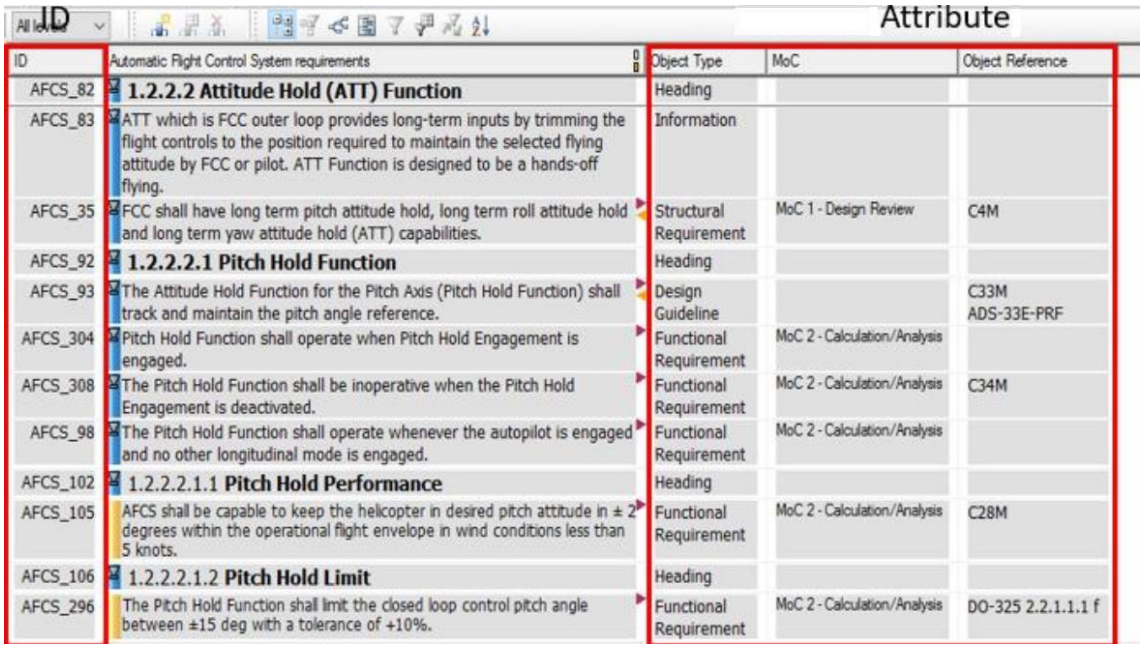


Figure 3. Requirement hierarchy

Unique ID assignment is made to each requirement in the requirement set. In addition, attributes that define requirements specifically are created for requirement sets. Object type means of complies (MoC) and requirement source / reference attributes were defined for system requirements.

- Object type refers to type of the requirements according to its content. The created object types are Information, Heading, Design Guideline, Structural Requirements and Functional Requirements.
- Means of compliance (MoC), expresses with which method to validate the requirements. The created MoC are Compliance Statement, Design Review, Calculation/Analysis, Safety Assessment, Laboratory Tests, Ground Tests, Flight Tests, Design Inspection/Audit, Simulation and Equipment Qualification.
- Object Reference refers to the source of the created requirements. The created object references are Engineering Judgment, Contractual Requirements and Standards.

An example of the ID and attributes defined in the requirements is as in Figure 4.



ID	Automatic Flight Control System requirements	Object Type	MoC	Object Reference
AFCS_82	**1.2.2.2 Attitude Hold (ATT) Function**	Heading		
AFCS_83	ATT which is FCC outer loop provides long-term inputs by trimming the flight controls to the position required to maintain the selected flying attitude by FCC or pilot. ATT Function is designed to be a hands-off flying.	Information		
AFCS_35	FCC shall have long term pitch attitude hold, long term roll attitude hold and long term yaw attitude hold (ATT) capabilities.	Structural Requirement	MoC 1 - Design Review	C4M
AFCS_92	**1.2.2.2.1 Pitch Hold Function**	Heading		
AFCS_93	The Attitude Hold Function for the Pitch Axis (Pitch Hold Function) shall track and maintain the pitch angle reference.	Design Guideline		C33M ADS-33E-PRF
AFCS_304	Pitch Hold Function shall operate when Pitch Hold Engagement is engaged.	Functional Requirement	MoC 2 - Calculation/Analysis	
AFCS_308	The Pitch Hold Function shall be inoperative when the Pitch Hold Engagement is deactivated.	Functional Requirement	MoC 2 - Calculation/Analysis	C34M
AFCS_98	The Pitch Hold Function shall operate whenever the autopilot is engaged and no other longitudinal mode is engaged.	Functional Requirement	MoC 2 - Calculation/Analysis	
AFCS_102	1.2.2.2.1.1 **Pitch Hold Performance**	Heading		
AFCS_105	AFCS shall be capable to keep the helicopter in desired pitch attitude in ± 2 degrees within the operational flight envelope in wind conditions less than 5 knots.	Functional Requirement	MoC 2 - Calculation/Analysis	C28M
AFCS_106	1.2.2.2.1.2 **Pitch Hold Limit**	Heading		
AFCS_296	The Pitch Hold Function shall limit the closed loop control pitch angle between ±15 deg with a tolerance of +10%.	Functional Requirement	MoC 2 - Calculation/Analysis	DO-325 2.2.1.1.1 f

Figure 4. Requirements in DOORS with attribute columns

3.1.2. Validation of Requirements

It indicates that the requirements are complete and correct. Validation process is usually done using a checklist of requirements. Requirements are updated by considering the missing and inaccurate statements resulting from the checklist and analysis.

Table 1. Example of requirement validation checks [7]

No	Correctness Checklist
1	Is it identifiable as a requirement?
2	Is the requirement redundant?
3	Does the requirement conflict with others?
4	Is it physically possible to meet the requirements?
5	Is the requirement set better suited to be combined into a single requirement?

3.1.3. Validation of Requirements

Traceability in requirements defines the whole life process of requirements. The life process of requirements starts from where its history and source are based and continues to new requirements that will be created throughout the development period. The requirement set with more general expressions is defined as the highest level, and the requirement set with all the details to design a system is defined as the lowest level. Traceability between requirements occurs when a lower level requirement meets a higher-level requirement. In this case study, traceability has been provided between the lowest level software requirement set and the highest-level customer requirement set with the connections. Traceability of the requirements is very important for verifying the requirements. Verification of requirements are defined as demonstrating that the system is designed correctly according to customer requirements as a result of implementation of the requirements. The requirement validation process begins after the design is finished and checks that the design has been made in accordance with the requirements.

There are several methods (MoC) for requirement verification. Some of the requirement validation methods are shown in Table 2.

Table 2. Means of compliances

MoC Code	MoC Description	Associated Compliance Documents	Definition
MoC 1	Design Review	Descriptions Drawings	Compliance is proven by the design review minutes, system description documents, drawings, etc.
MoC 2	Calculation and Analysis	Substantiation reports	Compliance is proven by an analysis activity and report, such as static and fatigue strength analysis, load analysis, platform performance analysis, off-line simulation modelling analysis etc
MoC 3	Safety Assessment	Safety analysis	Compliance is proven by reference to the safety documentation defined in Safety Program Plan.
MoC 4	Laboratory tests	Test programs Test reports	Compliance is proven by tests done on i.e. a specific rig test, subsystem bench test or system integration test activity

Before the verification of the requirements, test scenarios are created according the requirements. The models developed at the design stage are tested to verify the requirements. If the system features in requirements are satisfied completely in the test results, requirements are considered verified.

3.1.4. Transfer and Synchronization of Requirements

By transfer and synchronization between the IBM Rational Rhapsody that is created models of requirements and "DOORS" that is managed of the requirements were provided to continuous integration between requirements and models. As shown in Figure 5, integration is provided by using IBM Rational Rhapsody Gateway add on.

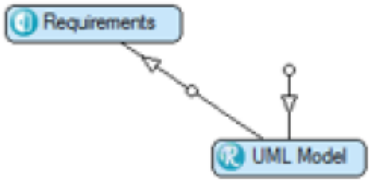


Figure 5. Requirements and model connection in IBM Rational Rhapsody Gateway

3.2. System Architecture Management

System architecture management is carried out with SysML which is a visual / graphic based architectural modelling language used in systems engineering applications. SysML has a grammar and vocabulary just like any of the natural languages we speak in this World (ex. English, Japanese etc.) [5]. Models are created to develop system architectures with SysML.

SysML models are examined under three main titles structural, behavioural and requirement. Various diagrams are used to create SysML models. SysML diagrams are shown in Figure 6.

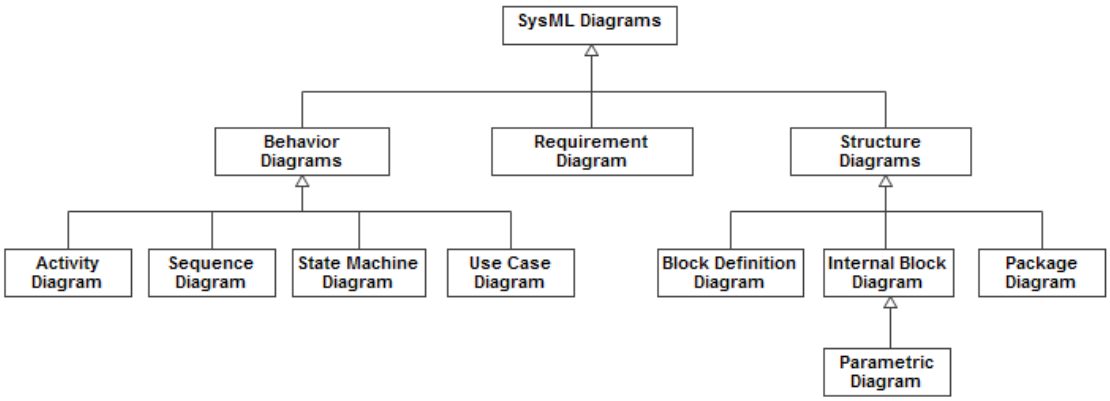


Figure 6. SysML taxonomy

The structural expression of a model answers the question: What is the system? Structural models are used to

- define the system,
- define system components,
- define system features,
- define system constraints,
- define the system model organization,
- determine their behaviour,
- define the relationships between system elements.

The behavioural expression of a model answers the question: How is the system behaves? Behavioural models are used to

- define the behaviour of the system,
- define use case of the system,
- define functions of system,
- define activities of system,
- define sequence of behaviour,
- define operations within the system elements.

Within the context of the case study, a hierarchy was created using SysML that contains the packages related to the its contents. The hierarchy shown in Figure 7 is the model organization of the system and was created using packages.

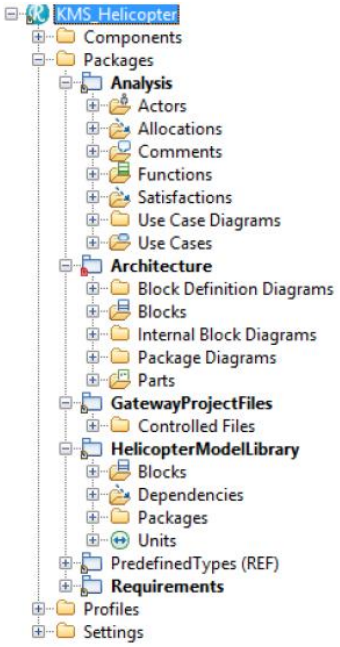


Figure 7. Case study model organization

System architecture management is provided by this model organization. Model organization was created with 6 different packages:

- The "Analysis" package consists of behavioural models and requirement analysis including Use Case Diagrams, Activity Diagrams, Sequence Diagrams.
- The "Architecture" package consists of structural models containing Block Definition Diagrams, Internal Block Diagrams and Package Diagrams.
- The “GatewayProjectFiles” package, consists of IBM Rational Rhapsody Gateway project files that provide communication between DOORS software used in requirement management and IBM Rational Rhapsody software where the system architecture is realized, and which contain various relationships between each other.
- The “HelicopterModelLibrary” package is a library containing the model elements, relationships, units and sub-packages used in the study.
- The "PredefinedTypes" package consists of models containing stereotypes representing the features to be used in the case study. It is defined automatically when the SysML project is created.
- The "Requirements" package includes models of requirements created within the scope of the case study and requirements found in the DOORS software.

3.3. Model-Based Systems Engineering Processes

Model-based systems Engineering (MBSE) is an approach used to reveal the needs that involve different perspectives of stakeholders, which one of the problems of systems engineering, and to analyse these requirements with models and make them more detailed and understandable. Thus, in complex systems like a helicopter, the entire process from customer requirements to system requirements and system architecture can be explained with this approach. In this process, requirements analysis, functional analysis, system architecture design and detailed system architecture design phases are carried out respectively.

3.3.1. Requirement Analysis

The requirements analysis process is the first phase of the model-based systems engineering process. The input of the requirements analysis process is the customer requirements, and the output is the use case models of the system. The requirements analysis process includes the design steps listed below:

- Examining customer requirements and determining required behaviours (use case)
- Derivation of preliminary system requirements for the use case determined with reference to customer requirements and standards
- Linking the derived system requirements to customer requirements.

Models are used when expressing targeted behaviours (use case) in the model-based systems engineering process. In SysML, the use case that is created with using phrases and actors of the system are expressed with the Use Case Diagram. After classification of customer requirements, use cases and actors are built in use case diagrams shown in Figure 8 to link the related requirements. It is intended to cover all customer requirements during the use case definition process. In addition to the actors and system boundary have been defined.

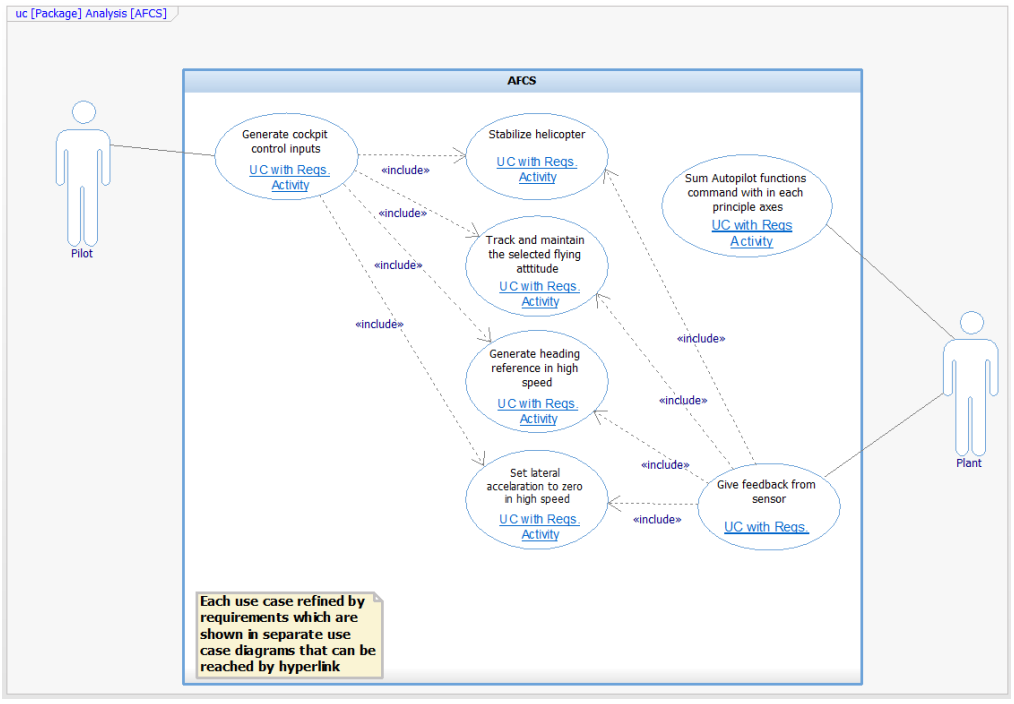


Figure 8. Use Case Diagram example from case study

After creating the success and establishing connections, preliminary system requirements started to be produced. The system requirements derived for “stabilized helicopter” use case and the “refine” connection between use case and system requirements in the Use Case Diagrams are linked as shown in Figure 9. Thus, system requirements have been created for the relevant use case.

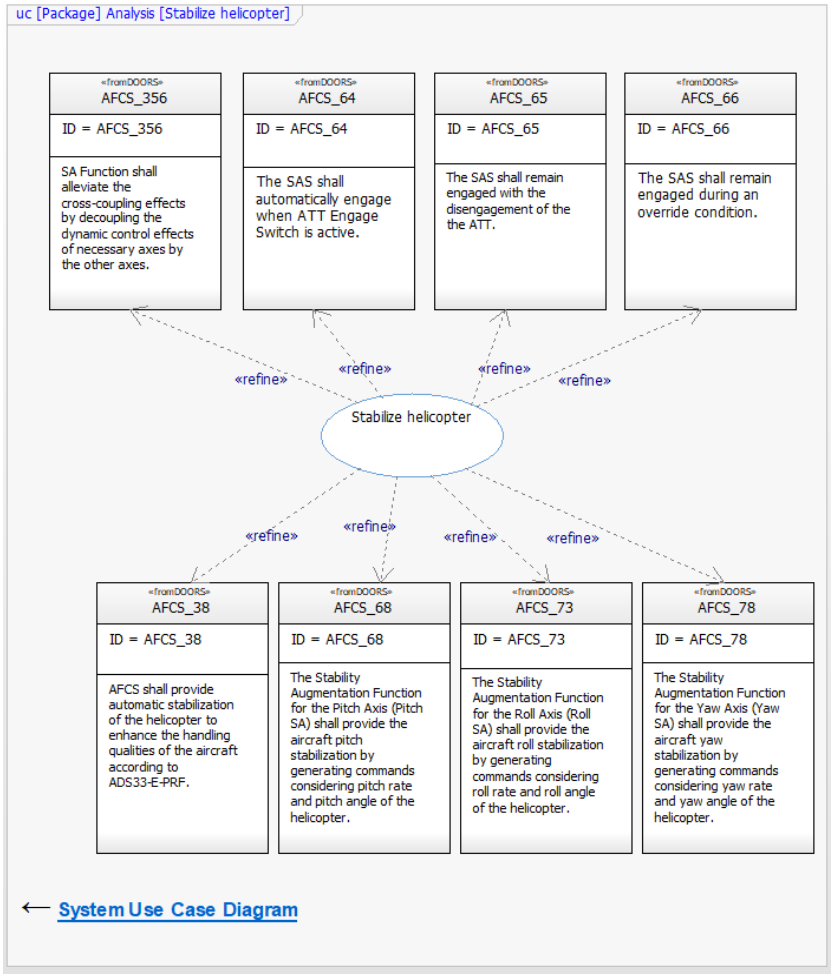


Figure 9. Use Case Diagram example with refine dependency

3.3.2. System Functional Analysis

The functional analysis process is the second phase of the Model-based System Engineering process. Functional analysis is defined as a systematic process for defining and associating the functions that a system must perform in order to be successful. The functional analysis process is carried out for the following steps.

- To define all the functions that the system must fulfil to meet the requirements in a graphical model.
- Allocation of detail requirements created as a result of detailing system requirements
- Defining sub-functions required for each function by making functional decomposition
- Explain what to do and how to do it before implementing the requirements.

The steps in the functional analysis process are shown in the Figure 10. [6]

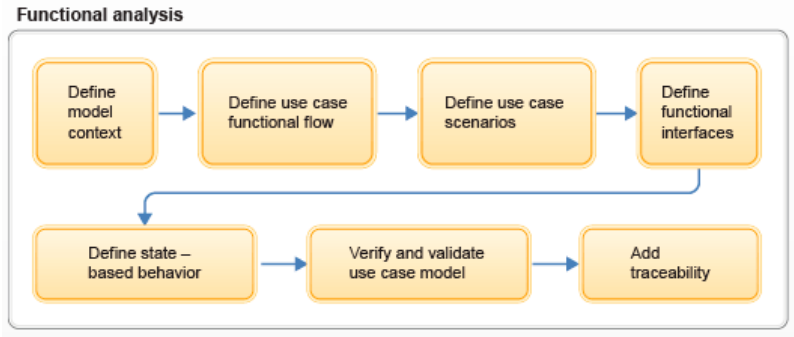


Figure 10. Functional analysis process

In the functional analysis process, which is one of the model-based systems engineering processes, the definition of the model content is determined by the use case. The creation of the detailed use case functional flow is provided by Activity Diagrams. While creating activity diagrams, actions are determined firstly, and control flow is obtained by taking into account the order of realization of these actions.

In the process of defining the scenarios of the use case, it is determined how a system can perform its use cases and whether there are any conflicts or conflicts between the lower level functions (actions) while performing these behaviours.

In the process of defining functional interfaces, the basic interactions between the system and the environment and the interconnections of the behaviours of the system components are defined. The inputs containing the data received from the outside of the actions and the outputs containing the data given out are determined.

Activity Diagrams created for the realization of the use cases that are desired to be in the system during the verification and validation of the use case model are verified by animations.

In the process of ensuring traceability, which is the final stage of functional analysis, it is shown that all requirements are covered by connecting requirements with Use case diagram components. The connection relationship is provided by “satisfy”.

The Activity Diagram shows the dynamic aspects of a system and the action-to-action control flow. It defines the basic interactions between the system and the environment, or the interconnections of the behaviour of the components. An Activity Diagram allows you to accurately transfer the most complex behavioural goals by creating different scenarios with various types of action. The "SAS" Activity Diagram created for the "stabilized helicopter" use case is as shown in Figure 11 with the data flow.

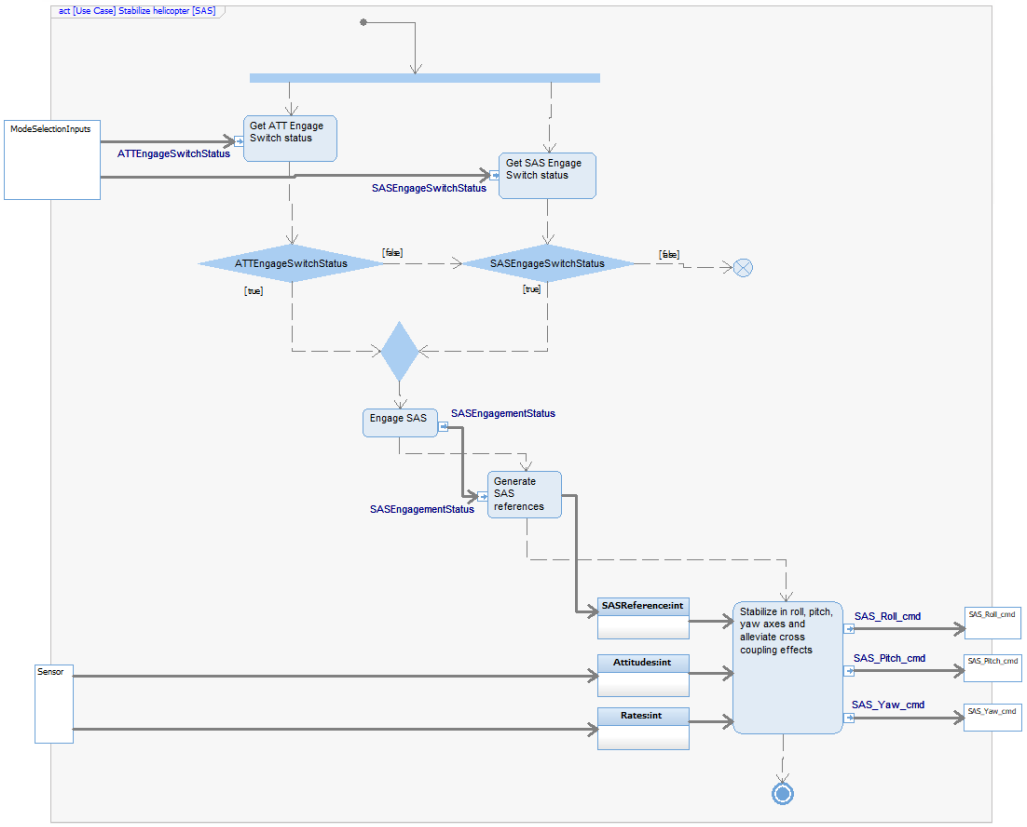


Figure 11. Activity Diagram example from case study

As a result of the creation of Activity Diagrams, the big picture of the behaviour of the helicopter automatic flight control system is revealed. The output of the functional analysis process is considered to be the creation of the functional architecture of the system and its definition of the functionality of the system. According to the ARP4754A document, the output of this process is defined as the determination of the scenarios and actions required for the realization of the use case of the system and the resulting functional requirements.[7]

Functional requirements define the functional infrastructure of the system, specify what the system will do in detail, express the necessary characteristics of the system and the constraints in the system solution. By obtaining functional requirements, the system requirements resulting from the requirement analysis are revised. Some of the functional requirements that are the output of the functional analysis process are as shown in Figure 12.

ID	Automatic Flight Control System requirements	Object Type
AFCS_18	SAS Engage Switch shall be a manual ON/OFF switch to establish engagement of SA Functions.	Functional Requirement
AFCS_348	The SAS Engage Switch shall remain its status when the Trim Release Button is pressed or released.	Functional Requirement
AFCS_251	The output commands of Autopilot Functions shall be summed separately for each principal control axis.	Functional Requirement
AFCS_70	Except where otherwise specified, a damping ratio of at least 0.3 critical shall be provided for nonstructural AFCF controlled mode responses. Specified damping requirements apply only to the response characteristics for perturbations an order of magnitude greater than the allowable residual oscillation.	Functional Requirement
AFCS_292	The Pitch SA Function shall limit the closed loop control pitch angle between ±15 deg with a tolerance of +10%.	Functional Requirement

Figure 12. Functional requirements example from case study on DOORS

3.3.3. System Architecture Design

The system architecture design process is the third phase of the Model-based systems engineering process. System architecture is defined as the conceptual model that defines the structure, behaviour and formality of the system.[8] In the system architecture design process, functional requirements are classified, a structural model component specific to each class is created, and functions are allocated to structural model components. Structural architectural and structural requirements obtain as a result of this process. While performing the system architecture design process in the case study, the path as follows:

- Defining basic system functions
- Classification of functions and creation of functional architecture
- Creation of structural components from functional architecture
- Allocation of system level operations to structural model components as shown in Figure
- Creation of structural architecture
- Obtaining structural requirements

The system architecture design process focuses on the development of a structural architecture that can perform the necessary functions within the limits of the estimated performance constraints. Structural models created in the system architecture design process,

- show which parts of the system will consist,
- show what the relationships between the parts will be,
- define the details / features of the internal structure of the parts,
- create the structural architecture of the system in a hierarchically.

Block Definition Diagram and Internal Block Diagram are created with SysML to define a structural architecture. In this case study, the structural architecture of the system was created hierarchically with different Block Definition Diagrams. Within the scope of this article, the design of the “SAS” system architecture was handled step by step by taking the "Stabilized helicopter" use case and the "SAS Activity Diagram" created during the functional analysis process.

Block	Allocation	Action
SAFunction	SAFunction	generateSASCmd
SASEngagement	SASEngagement	engageSAS
SASEngagement	SASEngagement	getSASEngageSwitchStatus
SASEngageSwitch	SASEngageSwitch	getSASEngageSwitchStatus
SASEngageSwitch	SASEngageSwitch	checkSASEngageSwitchStatus
SASReference	SASReference	generateSASReferences

Figure 13. Block and action allocation table

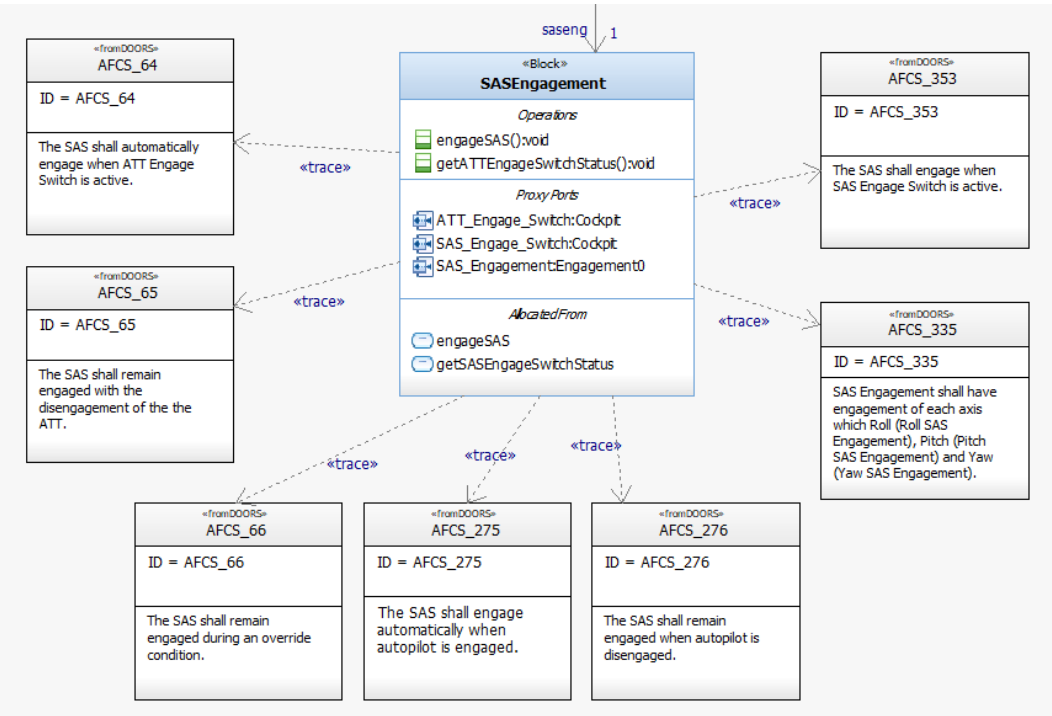


Figure 14. SAS Block definition diagram

It provides a visual representation to manage architectural system complexity and create a communication and coordination mechanism between components. The output of the system architecture process is conceptual models that define the structure, behaviour and formality of the system. These models are designed with Block Definition Diagrams. As a result of the creation of Block Definition Diagrams, the structural architecture and structural requirements of the helicopter automatic flight control system were revealed. With the obtain of structural requirements, the system requirements that obtained as a result of the needs analysis were revised.

Some of the structural requirements that are the output of the system architecture design process are as shown in Figure 15.

ID	Automatic Flight Control System requirements	Object Type
AFCS_2	AFCS shall consist of Cockpit Control Inputs, Flight Control Computer and Sensors models.	Structural Requirement
AFCS_6	Cockpit Control Inputs shall consist of Flight Control Inputs and Mode Selection Inputs.	Structural Requirement
AFCS_14	Mode Selection Inputs shall consist of switches/buttons (etc.) for each autopilot modes engagement.	Structural Requirement
AFCS_33	FCC shall provides helicopter flight control system that includes Autopilot Functions and Autopilot Logics.	Structural Requirement
AFCS_37	Autopilot Functions shall have Stability Augmentation (SA), Attitude Hold (ATT), Roll Heading Hold, Side Slip Functions.	Structural Requirement
AFCS_34	FCC shall have short term pitch, short term roll and short term yaw stability augmentation (SA) in all flight regimes.	Structural Requirement
AFCS_335	SAS Engagement shall have engagement of each axis which Roll (Roll SAS Engagement), Pitch (Pitch SAS Engagement) and Yaw (Yaw SAS Engagement).	Structural Requirement

Figure 15. Structural requirements example from case study on DOORS

3.3.4. Detailed System Architecture Design

The final stage of the process of Model-based systems engineering is the detailing of the system architecture. The detailing of the system architecture aims to explain in which order the structural architectural components of the system operate in accordance with the scenarios and to show the communication between the components. The system architecture developed using SysML is detailed with Sequence Diagrams. The process of obtaining Sequence Diagram in SysML is as shown in Figure 16. [9]

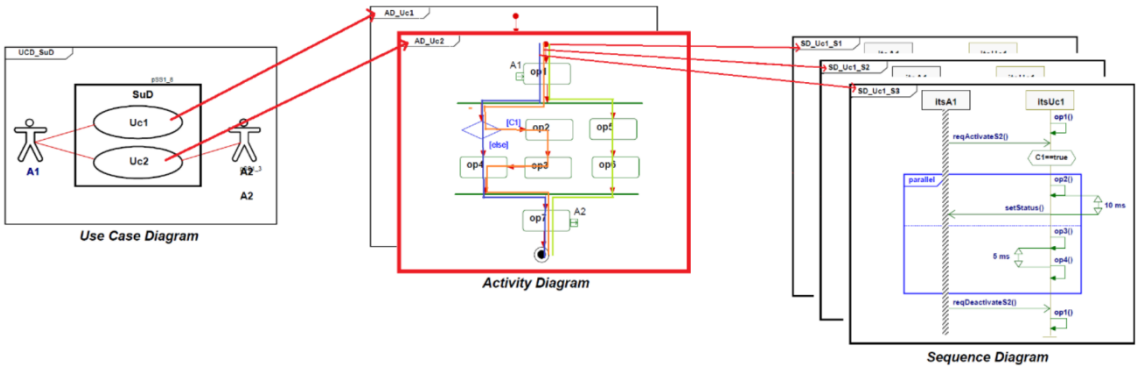


Figure 16. Process of the obtaining Sequence Diagram

When creating Sequence Diagram, a scenario is selected firstly from Activity diagram. The blocks in which the actions that are active in the selected scenario are allocated are added to the Sequence Diagram as a "lifeline" model component. Communication between the "Lifeline" model components is provided by messages. The detailing process of the system architecture was carried out in all the Activity Diagrams that the output of the system functional analysis in the case study. As an example of the system architecture detailing process, the helicopter automatic flight control subsystem “SAS” is detailed with the Sequence Diagram as shown in Figure 17.

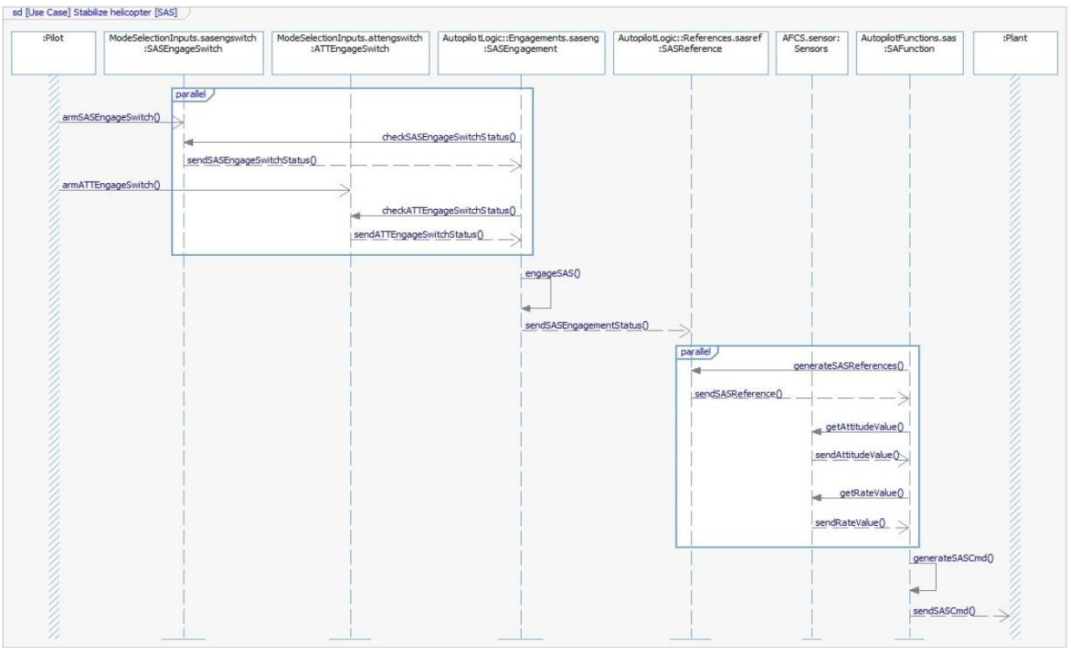


Figure 17. Sequence Diagram example from case study

During the system detailing process, the messages transmitted between the blocks in Sequence Diagrams are examined and the interfaces of the subsystems are created. The interfaces are shown in Internal Block Diagrams in SysML. The interface has been defined for all blocks created in the system. The helicopter and AFCS system interface to be used in the design is shown in Figure 18.

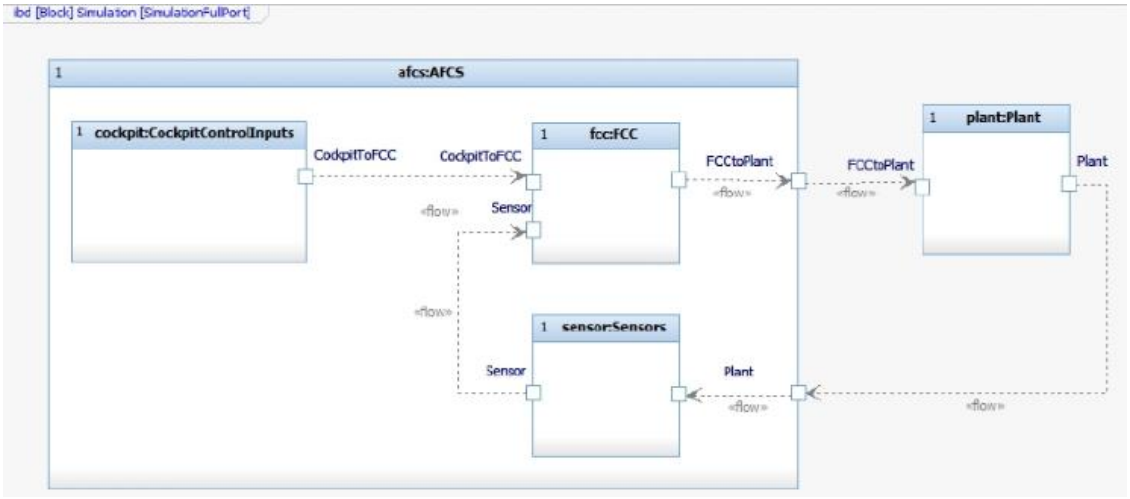


Figure 18. Internal Block Diagram example from case study

Interface blocks of helicopter automatic flight control system is shown Figure 19.

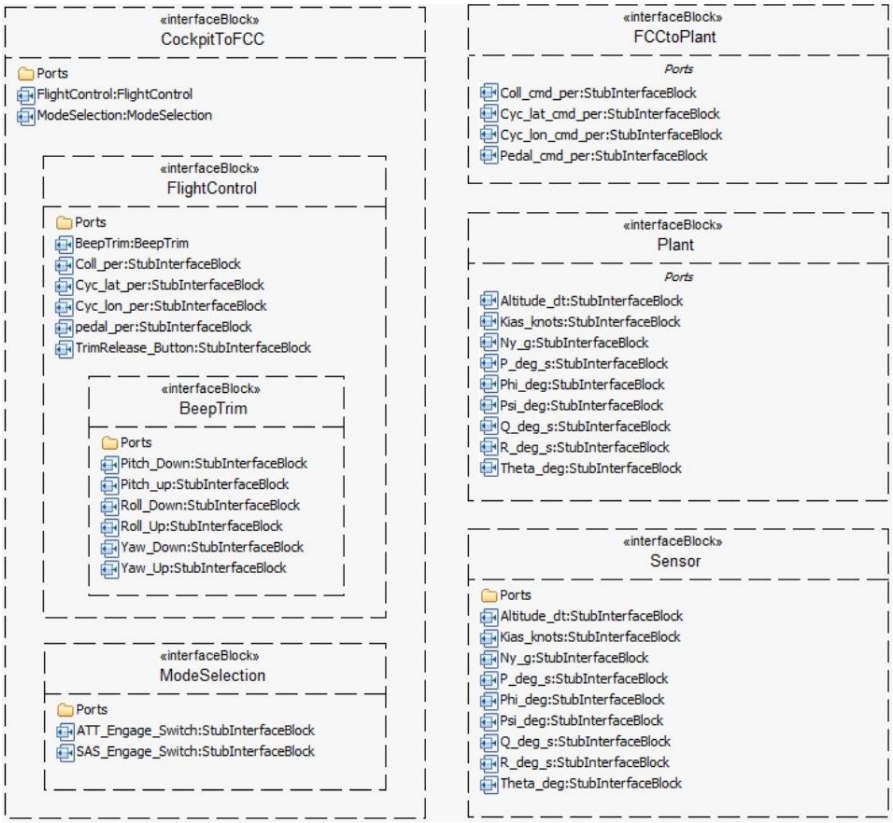


Figure 19. Interface block of AFCS case study

4. CONCLUSION

The model-based systems engineering process covers the requirements and system architecture stages of model-based design processes. Accordingly, requirements management, system architecture management and model-based systems engineering design processes have been developed.

Within the scope of the requirements management study, the requirements were defined in the DOORS software, a checklist was created for the validation of the requirements, the requirement was traceable, the requirement validation methods were defined, and the transfer methods were developed to use the requirements in different software.

Within the scope of the system architecture management study, the model organization for the helicopter automatic flight control system was realized in IBM Rational Rhapsody software with using SysML.

The following were found in the model-based systems engineering design processes.

- As a first stage the requirement analysis study, use case were revealed by reference to the customer requirements and the requirements were associated with the use cases.
- As a second stage the system functional analysis study, functional requirements of the system were revealed by developing functional architecture and functional models.
- Within the scope of the system architecture design study, structural requirements of the system were revealed by developing structural architecture and structural models.
- Within the scope of the detailed system architecture design study, the system's operating scenarios and system interfaces have been created.

With the model-based systems engineering approach and application of this study, a solution was found to the main problems of traditional system engineering. Model-based systems engineering approach is more systematic than traditional systems engineering but requires more preparation before implementation. As a result, a case study has shown that it is a more efficient design process for management and traceability.

5. FUTURE WORKS & LIMITATION

Similar to the work done in this article in the future, it can be applied in all aircraft design processes. More detailed testing and verification can be done using an advanced simulation infrastructure program. Traceability can be achieved by providing integration between programs where designs in different disciplines are realized and systems engineering designs.

As a limitation of the study, according to the methodology applied in this study, the requirements and system architecture stages of the model-based design stages are carried out for the automatic flight control system.

ACKNOWLEDGMENT

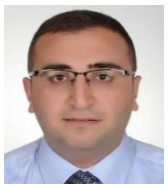
The material is based upon work supported by Turkish Aerospace Modelling and Simulation department. The authors would like to thank modelling and simulation co-workers for them supports.

REFERENCES

[1] INCOSE, Systems Engineering Handbook, INCOSE, 2004.
[2] S. Team, Systems Engineering Manual, FAA, 2014.
[3] Incose, Systems Engineering Vision 2020, Incose, 2007.
[4] Karagoz, Esma & Reilley, Kevin & Mavris, Dimitri. (2019). Model-Based Approach to the Requirements Analysis for a Conceptual Aircraft Sizing and Synthesis Problem. 10.2514/6.2019-0498.
[5] S. Friedenthal, A. Moore, and R. Steiner.OMG Systems Modelling Language Tutorial, 2009.
[6] IBM, "IBM Knowladge Center," IBM, [Online]. Available: https://www.ibm.com/support/knowledgecenter/SSB2MU_8.3.0/com.ibm.rhp.sysml.doc/topics/rhp_c_functional_analysis.html. [Accessed 5 September 2020].
[7] SAE, "ARP4754 - Guidelines for Development of Civil Aircraft and Systems," SAE, 2010.
[8] S. P. J. Holt, SysML for Systems Engineering 2nd Edition, 2013.
[9] H. Hoffmann, System Engineering Best Practices with the Rational Solution for Systems and Software Engineering, IBM, 2009.

AUTHORS

Haluk Altay



He is a graduate of Yıldız Technical University Mechatronics Engineering addition a student of M.Sc Istanbul Technical Universty Mechatronics Engineering. He has been involved in academic research for the past two years and worked several projects for the past three on modelling, flight mechanics and controls. He has experienced in Model Based Systems Engineering, Flight Mechanics and Dynamics, Aircraft Design, System Identification at Turkish Aerospace.

Muhammed Furkan Solmazgül

Furkan Solmazgül has been involved in model-based systems engineering projects since 2018. He is currently working as a design engineer at Turkish Aerospace. He graduated from Istanbul Commerce University in Mechatronics Engineering.