An Approach to Designing PHM Systems with Systems Engineering

Thomas Dumargue[1], Jean-Robert Pougeon[2] and Jean-Rémi Massé[3]

[1,2,3] *Safran Aircraft Engines, Villaroche – 77550 Moissy-Cramayel, France*
thomas.dumargue@safrangroup.com
jean-robert.pougeon@safrangroup.com
jean-remi.masse@safrangroup.com

ABSTRACT

This work shows that given the numerous specific constraints that PHM systems for turbofan engines have to deal with and despite the low level of criticality of such systems, the use of Systems Engineering (or even System of Systems Engineering) methods and tools is essential to ensure project success. Various aspects of the methodology which were applied in our projects are presented here: on one hand general system design and project management considerations, on the other hand transversal methodological items such as model-based systems engineering and methods to manage requirements, hypotheses, interfaces, configuration, change, compatibility, validation, verification and integration. Associated pitfalls and lessons learned from applying these elements are highlighted, especially the importance of defining from the beginning of the design all the project management plans, with a constant focus on customer needs satisfaction, interfaces and design documentation while allowing iterations.

1. INTRODUCTION

Over the last decade, Safran Aircraft Engines has been developing PHM functions and systems to improve its customer services around turbofan engines. The first iterations were based on existing engine fleets, however we are now building functions and systems for engines which are currently being developed. This brings more project constraints as there are more interactions with the engine development and because the PHM system needs to be ready at the engine's entry into service.

As with complex systems, the design of PHM systems and their components requires the use of Systems Engineering methods to ensure a more robust and efficient design. However the practical application of these methods can sometimes lead to rework and ineffectiveness issues. In the context of the design of a PHM system for turbofan engines, a lot of additional constraints appear, including but not limited to: weight, cost, high number of stakeholders/actors, aircraft manufacturer and final user constraints, imposed engine hardware, low embedded computation power and memory, low aircraft-to-ground data transfer capability, data protection and intermediate processing by the aircraft and its manufacturer. These constraints bring needs for even more thorough interfaces definition, requirements management and configuration management, sometimes with Systems of Systems considerations. Among key aspects, compatibility, interoperability and hypotheses management are to be found, as well as model-based systems engineering.

PHM literature mainly deals with the technical aspects (e.g. diagnostics/prognostics techniques or maturation), and Systems Engineering (SE) is also widely discussed and has its own field literature. Those two worlds do cross sometimes to put PHM systems design into perspective and to provide methodology considerations fitting PHM specificities. Indeed several papers detail validation and verification approaches for PHM systems (Markosian, Feather, Brinza, & Figueroa, 2005; Feather & Markosian, 2006; Feather, Goebel, & Daigle, 2010; Roychoudhury, Saxena, Celaya, & Goebel, 2013) or adapted requirements processes (Saxena et al., 2010, 2012). There is even a SAE standard (ARP6883) in progress about guidelines for writing IVHM requirements for aerospace systems, which are basically described in (Rajamani et al., 2013). (Saxena, Roychoudhury, Wei, & Goebel, 2013) goes even further with process/program requirements and several other high-level SE considerations such as design, hypotheses and interfaces management.

However most of the time PHM literature does not address all SE aspects, e.g. configuration management, and when several SE topics are evoked they are usually only associated with questions to consider without proposed tools nor lessons learned in practical applications. Moreover some considerations about Integrated Vehicle Health Management (IVHM) that can be found in literature may not be completely applicable to our turbofan engine context as the PHM system is not part of the monitored engine but rather shares some components with it.

To fill that gap, we propose to give an overview of most of Systems Engineering aspects and lessons learned in applying the associated methods and tools in the design of a PHM system for turbofan engines, along with encountered pitfalls and other practical concerns in the context of industrial projects.

This paper firstly details the design context of PHM systems for turbofan engines, notably specific applicable constraints, and reminds some Systems Engineering basics. Then we focus design practices at the system and project level, with a major highlight on project management plans. Finally we explain for every SE topic what its principle is, how it was applied in our case and which lessons we learned from that application.

2. CONTEXT

2.1. Design Context of PHM Systems for Turbofan Engines

When designing a PHM system for turbofan engines, one has to take into consideration several specific aspects, like some mentioned in (Lamoureux, Massé, & Mechbal, 2012).

First of all, the main goal of the system will be to enable the company to schedule critical maintenance operations, thus avoiding events like Delays and Cancellations (D&C) or In-Flight ShutDowns (IFSD).

The PHM system will then act upon two majors axes:

- assisting troubleshooting and diagnosis by locating faulty components
- anticipating maintenance actions by predicting upcoming degradations.

It is important to note that the PHM system in our context only qualifies as an economical service for the customer and is not involved in engine certification or regulatory requirements such as safety or reliability aspects.

The global data flow that can be found in most cases is described in Figure 1.

As showed in that figure, a major specificity is that the system is split over three main systems: the engine (from sensors to embedded computing), the aircraft manufacturer's system (aircraft and ground system) and the engine manufacturer's ground PHM system. Therefore designing the PHM system for a turbofan engine implies designing an embedded part and a ground part of the system, both with strong interactions with and dependence on the external aircraft manufacturer's system.

That distribution brings about several constraints and considerations described in the following sections.

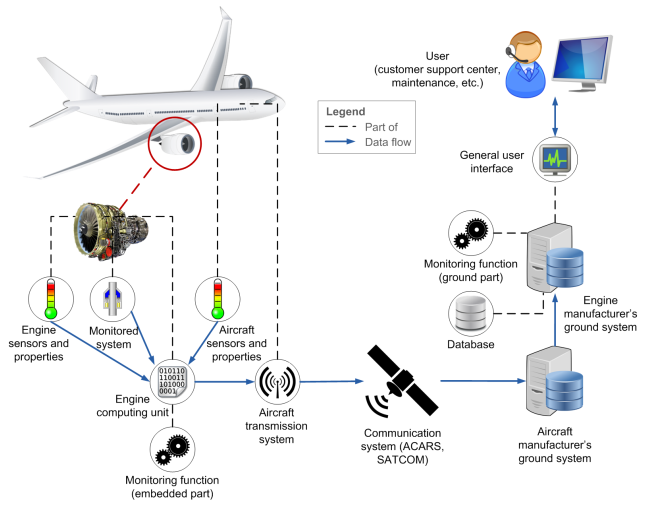

User
(customer support center, maintenance, etc.)
Legend
Part of
Data flow
General user interface
Monitoring function (ground part)
Engine manufacturer's ground system
Database
Engine sensors and properties
Monitored system
Aircraft sensors and properties
010110
110011
101000
0001
Engine computing unit
Aircraft transmission system
Communication system (ACARS, SATCOM)
Aircraft manufacturer's ground system
Monitoring function (embedded part)

Figure 1. Global data flow in a PHM system for turbofan engines

2.1.1. Engine-Related Constraints

Weight Weight is one of the key design drivers (KDDs) of a turbofan design, therefore every increase in the weight must be fully justified and profitable

Performance Engine performance is also related to KDDs (e.g. thrust and surge margin) and thus cannot be deteriorated by the inclusion of a PHM system

Imposed Engine Hardware Most of today's turbofan designs do not make much room for PHM considerations, whether because the engine was designed before the monitoring functions, because KDDs and engine certification aspects take over them or because design choices led to reuse some embedded components from previous turbofan designs. Then monitoring functions have to deal with imposed sensors (accuracy, location) or computation unit(s).

Low Embedded Computation Power and Memory Due to environmental, certification and engine constraints, computation units have very low computation power and memory available (the order of magnitude is similar to those of pocket calculators)

Cost Between the embedded and the ground parts of the PHM system, the embedded one is certainly the most expensive one. Indeed, its development, production and maintenance (computation unit, embeddable and certifiable code, integration) bring important costs, and so does every evolution or correction, even minor ones.

2.1.2. Aircraft Manufacturer Constraints

Low Aircraft-to-Ground Data Transfer Capability The aircraft-to-ground data transmission system is the aircraft manufacturers' responsibility, so the choice stands by them. Such currently-used systems (ACARS, SATCOM) imply costs directly proportional to the amount of data sent, which must therefore remain as little as possible. Authorized amounts for engine PHM data to be sent are incidentally specified by the aircraft manufacturer.

Intermediate Processing by the Aircraft and its Manufacturer Data is sent from the engine to the engine manufacturer's ground system through aircraft systems, a communication system (ACARS, SATCOM) and the aircraft manufacturer's ground system (see Figure 1). The associated protocols and formats are consequently mostly imposed and the engine PHM system has to deal with them.

2.1.3. Ground-System-Related Constraints

Host Platform Some architectural choices may lead to use an existing or common platform used for other applications and services than PHM to host the ground part of the PHM system. In that case additional constraints (e.g. language, protocols, memory usage, interfaces) can appear in the ground system design.

Multiprogram To avoid redesigning a complete PHM system for every turbofan program, the ground system is likely to be designed so that it can be compatible with several turbofan fleets, bringing again new constraints.

2.1.4. Other Considerations

Number of stakeholders The PHM system design involves several entities that are quite independent, among which can be found:

- the aircraft manufacturer
- the end-user (airline for instance)
- the engine manufacturer's customer support and maintenance teams
- the engine design teams
- the engine manufacturer's PHM teams

Ensuring a global consistency on the whole project requires thorough common methods, practices and framework.

End-user needs The end-users (whether internal or external) express needs that can sometimes bring additional constraints to the PHM system design.

Security The system must be invulnerable to external attacks, especially for the embedded part as it has links with the engine control system (FADEC).

Data Protection Engine PHM data are transmitted through several intermediate systems, making them potentially usable by competitors and thus raising the need to adopt an adequate strategy about how to keep most intellectual property results safe.

2.2. Systems Engineering

Systems Engineering (SE) is, as defined in (Haskins, 2010), *an interdisciplinary approach and means to enable the realization of successful systems*. The idea is to define methods and tools to do the right product (satisfying customer needs) and do the product right (functional and effective) while optimizing project aspects (quality, cost, time).

It covers the whole product lifecycle (e.g. research, development, production, maintenance) on a variety of fields such as requirements management, validation, verification, integration or configuration management by taking a global consideration of all these elements.

That approach takes its value from experience, based on the observation that most successful projects had focused a lot on such aspects, while most failed projects (e.g. cancelled or with cost or time overruns) had bad experience over those subjects, especially the lack of user involvement and incomplete requirements (The Standish Group, 1994).

In traditional design, projects focus less on the design phases than on production, integration and tests, leading to fix problems in the latter rather than in the former. Systems Engineering suggests spending more time and money on design to reduce risk for later phases, thus bringing fewer problems and rework and shortening those phases, allowing in the end to save time and money on the whole project. Beyond that intuitive approach, (Honour, 2013) provides a complete study on the Return-On-Investment (ROI) of the application of Systems Engineering in project. That thesis clearly shows the benefits of using such concepts but also raises the potential pitfall of spending too much effort on SE (overengineering).

Main guidelines for the application of SE in our context will refer to usually applicable standards (ARP4754A, 2010).

System of Systems Engineering (SoSE) Beyond Systems Engineering a new concept was created: the System of Systems (SoS), which can be defined as a gathering of independent systems that offer emergent functionalities once brought together. One famous SoS example is the Internet. All the methodology around Systems of Systems is called System of Systems Engineering (SoSE) (Jamshidi, 2008).

As of today SoSE is still a young field and the whole methodology is not completely defined yet and a lot of research is dedicated to it, the idea being to see to what extent SE can be transposed to SoSs and to define the potential missing methods and tools. That includes SoS architecting (Luzeaux, 2013) but also SoS monitoring (Shone, Shi, Merabti, & Kifayat, 2011) which involves for instance specific security issues. A prominent concern in that field is also the definition of standards, notably for interfaces design and data sharing (Fitzgerald, Larsen, & Woodcock, 2013).

3. DESIGN PRACTICES AT THE SYSTEM AND PROJECT LEVEL

The PHM system design must deal with the constraints described in section 2.1. At the system and project level, it is very important to place all the necessary foundations for a successful project: the earlier the work definition is settled, the better it will be as suggested by Systems Engineering (see section 2.2). These methods are particularly adapted to the design of such complex systems involving multiple components, design layers and stakeholders from different companies or departments if one wants in the end to ensure user needs satisfaction and the system consistency and reliability.

The main elements to define are described in the following sections.

3.1. Project Aspects

Project management plans (e.g. development plan, configuration management plan, risk management plan) detail the rules of the project, explaining how and when the work will be done. They are of utmost importance as they coordinate the work and practices over the whole project and its stakeholders and they are the reference used in case of conflict.

However as development usually starts after a research phase, they tend to appear quite late in projects, even after the beginning of the development phase. The later they are settled, the worse it gets as every team works in its own way. Planning activities and deliverables is of course necessary, but it is not enough: for instance work split (see “WBS” below), standards and tools must be defined before the work starts, even in research or preliminary work phase if possible in order to avoid rework or misunderstanding.

“Breakdown structures” are also key aspects:

- the documentation breakdown structure (DBS) presents all the documents involved in the project with their relationships
- the product breakdown structure (PBS) lists all the components of the system and their relationships
- the organization breakdown structure (OBS) defines the stakeholders, their roles, relationships and place in the company organization. It especially specifies who is in charge and make the decisions if needed.
- the work breakdown structure (WBS) splits all the activities and documents in the project over the project organization to clarify all the responsibilities and make sure every split activity is covered and done by only one entity.

The pitfall to avoid is to manage a PHM project applied to an engine program like a research project on the pretext that the monitoring functions cannot be fully defined and mature at the same pace as the engine project’s. This warning is particularly relevent in the context of a turbofan engine development as the engine program usually lasts several years, thus easily allowing time drifts for the associated PHM project. Not setting that PHM project’s foundations on time only increases risk and will at some point cause rework or conflicts and result in time and cost overruns.

Moreover it is essential to consider planning aspects very early. The SE philosophy implies spending more time on the first phases of the project than on the late ones, although it can seem to be a risky bet especially when the project has tight deadlines. Its application must therefore be well thought out beforehand and consider also the fact that design teams may have different paces.

3.2. Relationship with the Turbofan Development

The PHM system design is intimately related to the one of the turbofan, as it monitors the engine components’ health and the engine includes elements serving the PHM purpose such as sensors or the computing unit (Massé, Lamoureux, & Boulet, 2011). It is therefore essential not to uncouple the design of both systems.

3.3. Customer/User Needs

As long as customer/user needs are not defined, hypotheses must at least be discussed with the customer. If not, design teams start working without even knowing if they are going the right way. These hypotheses must be associated with risks (according to the risk management plan) to be able to make appropriate decisions.

3.4. System Specification

One main aspect of SE is to elicit and analyse customer needs, in order to produce the system requirements. That document is the key deliverable in the project, as it covers the whole system from both an external and internal point of view. From an external point of view, it allows to make sure the customer needs have been fully covered and correctly understood and it can be used as a contract as requirements are non-ambiguous and verifiable. From an internal point of view, it settles the core requirements that will be developed at lower levels.

Defining indisputable requirements and expectations, notably between two independent parties (such as with a subcontractor), is one of the guarantees to avoid future conflicts or reworks.

When analysing customer needs, an operational analysis has to be clearly and exhaustively made, including use cases that show different situations and actors interacting with the system and how the system is expected to work in those cases. This operational analysis allows again to make sure both parties understand each other and form good roots for the product verification and the customer's acceptance tests.

As there are several articles in literature dealing with requirements like (Saxena et al., 2010, 2012, 2013; Rajamani et al., 2013) and the SAE standard ARP6883 in progress, we will not discuss that aspect in further details, other than providing lessons learned in section 4.2.

3.5. System Architecture

Once the system requirements (or at least the KDDs) are defined, candidate architectures of the system can be assessed, and needs derived from the system requirements can be allocated to the different subsystems or components. Then the design process (sections 3.3, 3.4 and 3.5) can be repeated at the subsystem level. The process may also be iterative at a given level to ensure agreement and to provide more flexibility for evolutions, because it is sometimes better to make several small iterations than to try to get everything right the first time.

The PHM system for a turbofan can be seen on certain aspects as a System of Systems as it results from the combination of the engine, the aircraft, the transmission system and the engine manufacturer's ground platform hosting the ground PHM system. Today all these systems are mostly designed and managed independently and they can be used independently as well (except for the engine/aircraft couple), nevertheless it is their gathering that makes the whole PHM system offer a monitoring function and service. However on some other aspects the PHM system is not a pure SoS. For instance some components are specified and designed by deriving the same upstream requirements, and some components like monitoring functions are specifically designed for that system.

Despite not being a true SoS, the PHM system can benefit from some of SoSE methods like standard interfaces definition, modelling approaches (Fitzgerald et al., 2013; Asan, Albrecht, & Bilgen, 2013) and simplification principles (Kopetz, 2015).

In our case such SoSE methods were not considered at the beginning of some PHM projects and the design, operational or managerial independence of the different involved systems was ignored. This led to misunderstandings, incompatibilities or stagnation on some parts of the PHM system. In fact we should have considered, as SoSE suggests, that we may not have control over everything (or even have control over very few elements) in the system and dealt with that constraint in our design approach.

3.6. Development Assurance Level

The SAE standard ARP4754A Development Assurance Level (DAL) corresponding to most components in a PHM system for turbofans is DAL E, which is the most permissive one: it basically allows not to follow any of the standard guidelines. However being allowed not to follow them does not mean that one should not follow them. Some projects may fall into that trap and abandon the use of SE methods. Experience shows that most of the time the project fails in that case (The Standish Group, 1994).

In our case company standards provide further guidelines that enable us to avoid such pitfalls.

4. APPLICATION OF SYSTEMS ENGINEERING TO THE DESIGN OF A PHM SYSTEM FOR TURBOFAN ENGINES

This section presents, for each methodological item :

- a brief reminder of the item's principles and goal
- how it was applied in the context of the design of a PHM system for turbofan engines
- lessons learned in that application.

4.1. Model-Based Systems Engineering

One solution to build a system architecture is to use a model of the system and its components and base the whole design around models rather than documents like textual specifications. This approach is known as Model-based systems engineering (MBSE). In the case of complex systems or SoSs several modelling approaches can be used (Estefan, 2008; Fitzgerald et al., 2013) and the standardized one is to use the Systems Modelling Language (SysML) defined in (OMG, 2015) and based on the Unified Modelling Language (UML), the latter being used mostly in software engineering.

The model has other advantages as:

- it can easily be understood
- it represents the system in different conditions (including dysfunctional cases)
- it can be simulated to validate its relevance and completeness.

We used SysML to model some layers of the PHM system in one of our projects, with the idea of allowing a possible reuse for future projects. A first pitfall appears here with the use of SysML and the freedom and wide range of possibilities it offers, which can rapidly confuse both the modeller and the model user if not standardized throughout the project. In our case we used the IBM Rational Rhapsody tool with a Snecma SysML profile (overlay) developed to fit the company's design needs.

Figures 2, 3 and 4 are examples of SysML diagrams describing the context and architecture of a monitoring function.

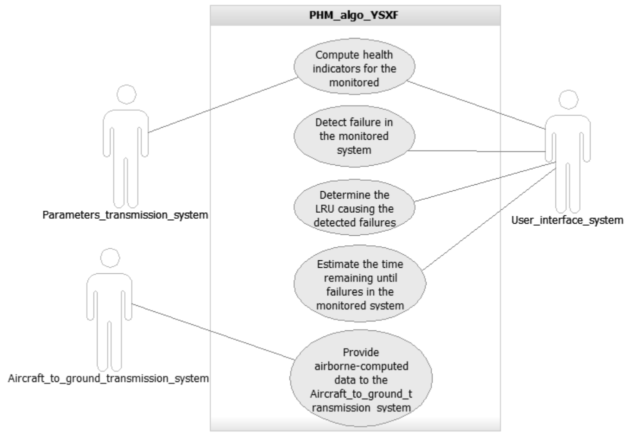

PHM_algo_YSXF
Compute health indicators for the monitored
Detect failure in the monitored system
Determine the LRU causing the detected failures
Estimate the time remaining until failures in the monitored system
Provide airborne-computed data to the Aircraft_to_ground_transmission_system
Parameters_transmission_system
User_interface_system
Aircraft_to_ground_transmission_system

Figure 2. Service diagram in SysML for a monitoring algorithmic function

Figure 2 shows the interactions and services provided by the monitoring function to the external stakeholders which are designated by "Parameters transmission system" (PTS, providing necessary information about the engine to the monitoring function), "Aircraft-to-ground transmission system" (AGTS, transmitting information from the embedded part of the monitoring function to the engine manufacturer's ground system") and "User interface system" (UIS, basically the engine manufacturer's ground system excluding the ground part of the monitoring function). This diagram allows to precisely define the considered system's perimeter and missions, i.e. its main functions. We can also see these elements from an interfaces point of view in Figure 4 with a larger context.

Figure 3 illustrates the high-level internal organic breakdown, flows and interfaces in the monitoring function, which can then be refined at a lower level. It shows by the way the use of the OSA-CBM layers (ISO 13374-1, 2003) in the architecture to have a standard approach between every monitoring function:

- the embedded algorithm, which includes part of the Data Acquisition (DA) and part of the Data Manipulation (DM) layers
- the Data Manipulation (DM) layer (which is actually here only the remainder of this layer with regard to the part in the embedded algorithm), which provides health indicators
- the State Detection (SD) layer, which provides indicators about the health state of the monitored system

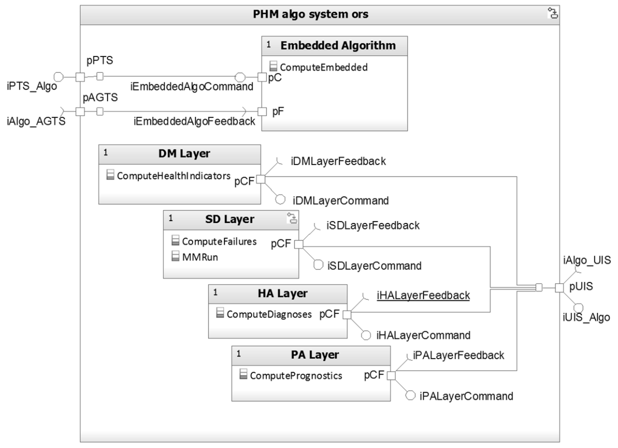


Figure 3. Organic internal diagram in SysML for a monitoring algorithmic function

- the Health Assessment (HA) layer, which provides diagnoses (faults or degradations identification and localization)
- the Prognostics Assessment (PA) layer, which provides prognoses about the system degradations

The Advisory Generation (AG) layer is not represented here at it is not part of the monitoring function described in the figure, nevertheless other components of the PHM system constitute that layer.

This approach allows to clarify even more the subsystems perimeter and interfaces between them and to make sure every design item has been considered. It also allows to contemplate the commonalities that can be set between PHM systems to optimize future developments and maintainability. One last purpose we considered was to be able to define accurate component specifications for subcontractors.

In our case, as the PHM system is not entirely part of the turbofan engine, the components shared between both systems have to be part of both system models too to ensure coherent design choices. This approach is highly facilitated by the use of a common framework (tools, rules).

However building and maintaining the model is a time-consuming activity and implies the involvement of every design team in the process, that is why there is still much to do on our side on that topic.

4.2. Requirements Management

As mentioned in section 3.4, requirements are the core elements of the system design, whether they are textual or not (model for instance).

In our case several issues appeared throughout one of our projects, the first one being that with the low required development assurance level (see section 3.6), no requirements

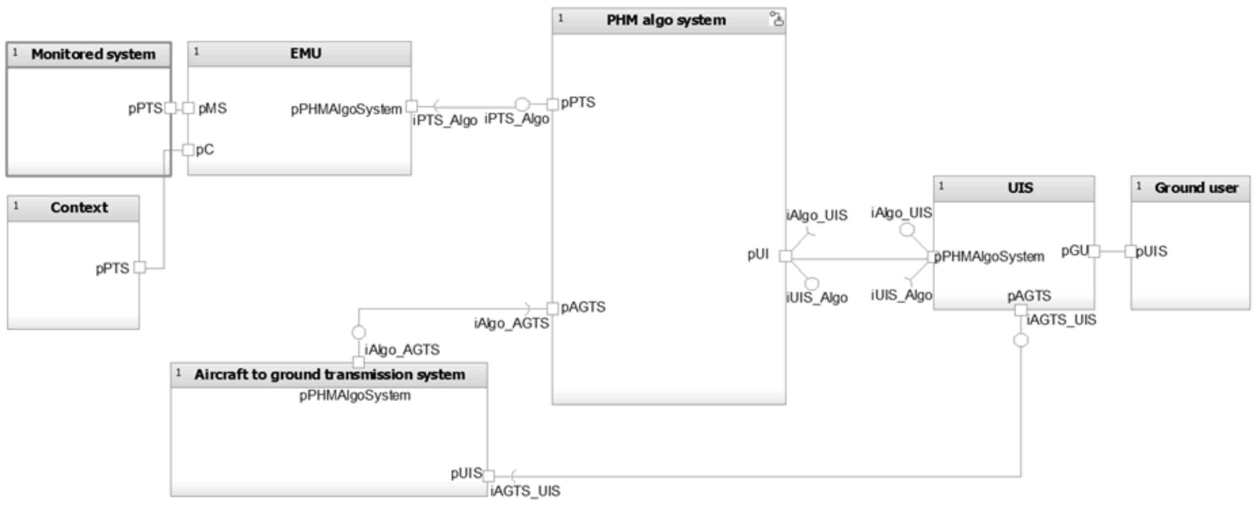


Figure 4. Context diagram in SysML for a monitoring algorithmic function

management rules were established, which led to other issues, such as:

1. the requirements at different design levels (especially the top-level ones) were not at the right level, nor clearly justified or traced to customer needs, nor verifiable
2. some requirements uselessly overconstrained the system design
3. some requirements were contradictory from one document to another (or even in the same document)
4. some requirements reference numbers appeared several times in the same document

In the end, after numerous iterations we had to start the specifications afresh to improve the situation. One major lesson learned here was to define a requirements management plan and stick to it, although this was not mandatory given the project DAL. The rules include that a system specification must define the system requirements from a black-box point of view, without intruding in the lower-level design. They also set guidelines for requirements wording (e.g. “the system shall”) and properties (e.g. uniqueness, atomicity and verifiability). These two rules allow to cover the first two problems listed above. We also introduced the use of a requirements management tool, IBM Rational DOORS, which was customized for our use and notably allowed us to impose specification structures, requirement attributes and the uniqueness of requirements reference numbers (fourth issue listed above).

Another aspect is the temptation to merge needs and requirements documents at the same level, usually when there is only one upstream needs document for a given product, because there is basically a one-to-one correspondence between requirements and needs (with additional requirements derived from expertise and experience). While this is totally acceptable in principle, it is not recommended when the parties writing these documents are different (e.g. not in the same team or not working at the same pace) because convergence is more difficult, non-compliances are not easily identified and it often leads to rework or misunderstanding.

A further consideration is the use of boilerplates (requirement structures) and ontologies (semantics) to guarantee even more that the requirements are consistent and well-written throughout the project, and allows to cover the third issue in the above list.

As for the model-based approach, there are impacts between the turbofan engine requirements and the PHM system requirements, through design constraints that one imply on the other (see section 2.1) and the fact that the shared components between both systems have to derive their requirements from both sides, with sometimes divergent interests (which is not the case when the PHM system is entirely part of the monitored system). Here again, the use of common tools and processes helped reducing risks on specifications.

4.3. Design and Hypotheses Management

Another consideration around the design phase is to formalise the design reflection and choices. Every requirement and design choice must be justified, for instance by experience, studies or tests and a record of that justification must be kept. But this is not enough: every contemplated design option and complementary analyses must also be written down so that future designers taking over the design has the necessary knowledge and no information is lost.

Also, every hypothesis must be clearly identified so that it can be properly managed regarding associated risks and validation actions.

In our case, much of the design information (including hypotheses) was not formalised or was disseminated in various locations (even designers' computers or emails) and there has been an important turnover in the design teams. As a result, a lot of research (and thus time and money spent) had to be done by the new designers and some pieces of information still remain unrecovered.

Therefore we set up design documents with all the information which was gathered and hypotheses management documents where the associated risks are identified, rated and combined with a risk reduction plan. We also wrote design practices with lessons learned in the knowledge management system for future designers. That allowed us to improve the validation level of our product and new members of the team were able to take over components more easily, which led to saved time and money in the project.

4.4. Interfaces Management

In systems architecture and integration, interfaces always represent a major issue, because most of the time they are the places causing the system to malfunction. That is why interfaces management is of utmost importance in the design.

First of all, the interfaces specifications must be known by both teams designing the interacting components. For that matter Interfaces Control Documents (ICDs) are good solutions as they can be used to specify the concerned subsystems as much as to integrate and verify the upper-level system. They result from the consultation of both subsystem designers, but it is the upper-level system's architect or integrator who is responsible for it. Thus we can make sure that every stakeholder involved around one interface agree and that every constraint has been taken into account.

In one of our projects, due to the different design paces between teams some interfaces were unilaterally defined and some constraints were passed on from the other side after the first product was issued. Besides incompatibilities and lacks of compliance, it led to rework which could have been avoided, had we considered interfaces as one of the core issue in the first place. Therefore for following versions of the system we included the interfaces definition in the first steps of the design to avoid those problems.

As mentioned earlier, in the case of turbofan engines, there are numerous interacting components, designed by separate teams or even companies, with several applicable constraints for each interaction, such as cost, bandwith, security or standardization (see section 2.1). Therefore it is of utmost importance here to restrict the number and complexity of interfaces as much as possible.

Indeed, the more complex and specific an interface is, the more hazardous it is. On the opposite, using as few and as standard interfaces as possible provides better chances for the project success. For instance in a data transmission interface it is easier to use a self-descriptive format and a standard protocol rather than building a specific protocol (for instance with a bit-by-bit data description) which is more subject to human errors and every evolution (e.g. new data transmitted) implies a modification of the ICD, the transmission system and the reception system. Sometimes interfaces cannot be as simplified as we would want them to be because of physical or economical reasons for instance, in which case there is still a need to be careful about the additional constraints raised by those interfaces.

In order to reduce the number and complexity of interfaces, Design Structure Matrices (DSMs) and associated tools can be used to define optimal modules in the system (Eppinger & Browning, 2012). The idea is to represent data and power flows between components in a matrix along with weighting factors translating for instance the intensity, risk or constraints associated with these flows. Then rows and columns can be switched with others to make the matrix as diagonal as possible and thus define optimal modules.

Figure 5 shows a simple example of such a DSM applied to a PHM system for turbofans which was built in one of our projects. Rows and columns represent the various components of the system (designated by numbers for readability), and the interface flows are here represented with arrows (there are no weighting factors in this case). We can see that modules have been constituted and the number of interfaces between these modules is limited. At this high level the interface description and module composition is quite easy and intuitive, but this kind of tools gets very interesting at lower levels, where there can be dozens of interfaces (including functional or software interfaces).

4.5. Configuration and Change Management

With the high number of components and stakeholders and the out-of-sync design paces between components, especially in our turbofan engine context where both the PHM system and the engine are not completely designed together, it is primordial to ensure that configuration (identification of all the components in the system and their versions) and evolutions are under control to guarantee the system consistency.

In our case, again because of the development assurance level (see section 3.6), no configuration or evolutions management plan was set in the project and no specific rules are followed. For instance some specifications (including ICDs) evolved without getting a new revision number, documents were not all stored in the same place, nor were the products, and some evolutions were made in documents or products without studying all the impacts on other parts of the PHM system. All these elements led to confusion and misunderstandings and then some necessary rework or incompatibilities.

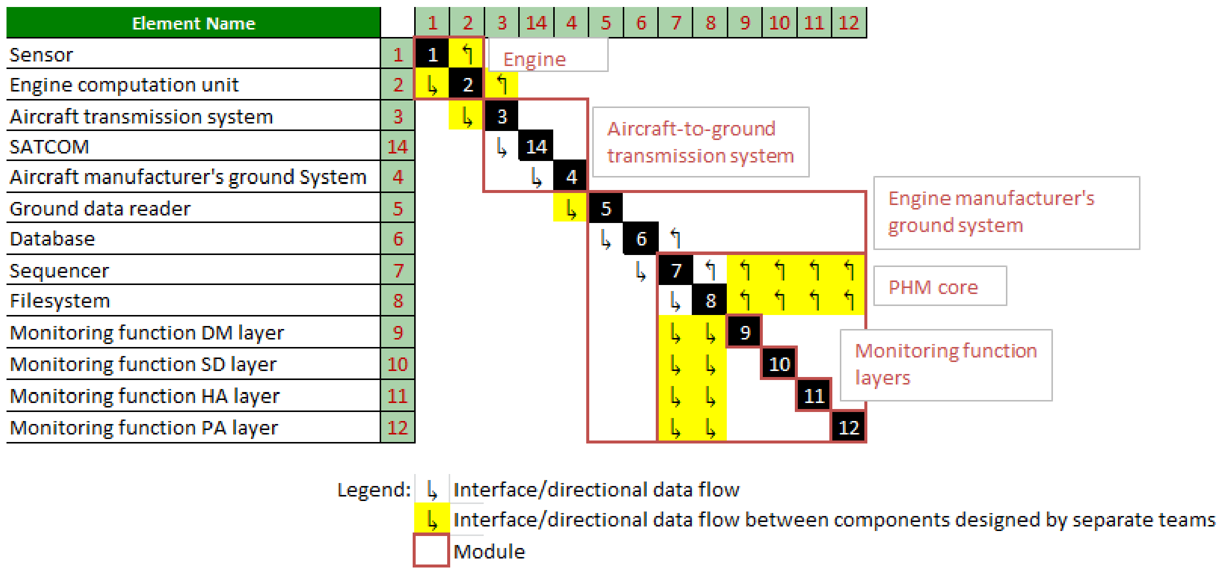

Element Name
Sensor
Engine computation unit
Aircraft transmission system
SATCOM
Aircraft manufacturer's ground System
Ground data reader
Database
Sequencer
Filesystem
Monitoring function DM layer
Monitoring function SD layer
Monitoring function HA layer
Monitoring function PA layer
Engine
Aircraft-to-ground transmission system
Engine manufacturer's ground system
PHM core
Monitoring function layers
Legend:
Interface/directional data flow
Interface/directional data flow between components designed by separate teams
Module

Figure 5. Example of Data Structure Matrix at the PHM system level

We improved the situation by establishing rules to manage configuration and evolutions. Among the rules can be found:

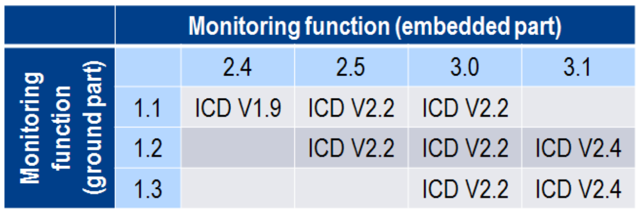
- every document in the project must be stored in the same shared repository (the same rule applies for products)
- every new version of a document must have a new revision
- system configurations are identified and versions of documents and products are associated to that configuration
- from the moment a consistent system configuration is identified, every document or product evolution must result from a common project decision after a complete impact analysis

Of course, as explained in section 2.2, a reasonable amount of SE must be spent on the project. In particular, clear rules about the level of configuration management, how configurations are defined and the conditions of evolution management must be established. For instance system configurations may not include all the lowest-level components but can be restricted to subsystem versions (or modules defined in DSMs), the lowest-level components versions being associated with the corresponding subsystem configuration/version.

4.6. Compatibility/Interoperability Management

Along with configuration management, one must also identify which versions of components are compatible and interoperable with one another. That issue is getting more and more complex as the number of versions for every component increases, which is why some tools can be used to manage compatibilities, such as the table shown in figure 6.

The choice can be made to ensure backward compatibility as much as possible, but that decision can lead to unnecessarily complicated products. Therefore backward compatibility must be justified due to the associated cost.



		Monitoring function (embedded part)			
Monitoring function (ground part)		2.4	2.5	3.0	3.1
	1.1	ICD V1.9	ICD V2.2	ICD V2.2	
	1.2		ICD V2.2	ICD V2.2	ICD V2.4
	1.3			ICD V2.2	ICD V2.4

Figure 6. Example of compatibility matrix

4.7. Validation and Compliance

Validation is also a main aspect of Systems Engineering as it covers all methods and tools aiming at satisfying customer needs.

Sometimes the focus on customer needs can be forgotten because of project constraints, and the lack of methodology and tools sometimes brought validation issues (absence of traceability or justification for example). It is always good to put things into perspective and think about the purpose. Also, consulting regularly the customer to check if what is designed corresponds to his expectations ensures we keep going the right way.

We applied several validation techniques, such as:

- requirements validation through traceability, justification and wording
- hypotheses validation and technology maturation based on studies, data analyses, tests, simulations or experience (Dupont & Massé, 2016; Massé, Hmad, & Boulet, 2012;

Lacaille, 2010)
- a compliance status regarding customer needs.

Here also a model of the system may help to validate it.

About hypotheses validation and technology maturation, the reader may also be interested in considering Technology Readiness Levels (TRLs) (Roychoudhury et al., 2013).

The compliance status is of utmost importance as it is the only way for the higher-level designer to know whether all allocated requirements are covered or not and thus make sure that nothing was forgotten in his different subsystems. It is useful to distinguish two kinds of compliance statuses:

- *a priori* statuses, used to indicate if we think we can be compliant based on the current hypotheses, usually at early stages, when the requirements flow-down is not over and the product has not been built
- *a posteriori* statuses, used to indicate if the product verification showed that the product meets the requirements associated to the concerned requirement.

A development and validation framework for health monitoring algorithms was also developed (Lacaille, 2009, 2012) and some validation tools were also created to automate parts of the validation process like test bench data analysis.

4.8. Verification

Although verification is one of the most intuitive aspects of Systems Engineering as it consists in checking that the product meets its requirements, it is not always well carried out. One of the tricky considerations concerns the expected verification level.

Indeed, as mentioned in section 3.6, the ARP4754A DAL E does not require a precise verification of the system, but not verifying anything would be hazardous. On the opposite, the engine is designed and certified under DAL A, which requires complete verification actions and procedures for every system requirement, like unit testing for software parts.

At first in some of our projects the decision was made not to set requirements or rules about verification, but in the end a few components were malfunctioning and the system as a whole was not working fully properly. That is why we can recommend that, even when having no obligation to make proper verification actions, one should at least test components by following typical use cases as in acceptance tests, although the test case data may not be the same as the customer's to provide a double-check. By setting up that kind of test cases we were able to find problems and fix them as soon as possible, which allowed us to save time and money on the whole project.

Also, appropriate verification processes should be considered depending on the system or component maturity level, as described in (Roychoudhury et al., 2013).

4.9. Integration

Integration is also a natural step in the PHM system lifecycle where components are brought together to make a (hopefully) functioning system or subsystem. Nevertheless this stage cannot solely rely on the fact that all the concerned components have been verified, because new problems may arise when interfacing them or when trying to run the whole system or subsystem.

We encountered such issues after some integration steps were skipped and we had to get back to them eventually: no gain, just pain. Hence it is very important to define integration-level actions, whether for design or verification. After introducing such actions in our processes like tests on the integrated ground system, we were able to identify where some problems were and to fix them at an earlier stage than before.

5. CONCLUSION

This work showed that there are a lot of constraints, components and stakeholders in the design of a PHM system for turbofan engines and such projects therefore need to apply the Systems Engineering (or even System-of-Systems Engineering) methodology to be successful.

Despite potential incompatibilities between project deadlines and the time that should be spent on Systems Engineering, we strongly recommend to consider applying as many of these methods as possible in projects as much time was spent on our side on rework or struggle although it could have been avoided with such methods applied. However there was sometimes an opposite tendency to go too far in the practice of Systems Engineering that led to overengineering the system. Hence it is a subtle balance that has to be found between not enough or too much SE, and it may require a certain flexibility in the design process so that it can evolve depending on the project needs.

Most SE methods have been successfully applied in our projects, some are still ongoing like modelling, but every lesson learned here (like the definition of all the project management plans at the beginning of the project and the need to constantly focus on customer needs, interfaces and design documentation) will be useful for future projects and new methods and tools will hopefully make them even more successful with shorter deadlines and a smaller cost.

REFERENCES

Asan, E., Albrecht, O., & Bilgen, S. (2013). Handling complexity in systems of systems projects – lessons learned from mbse efforts in border security projects. In M. Aiguier, F. Boulanger, D. Krob, & C. Marchal (Eds.), *Proceedings of the fourth international conference on complex systems design and management.*

Springer.

Dupont, A., & Massé, J. R. (2016, June). PHM functions maturation. In *Prognostics and Health Management (PHM), 2016 IEEE Conference on.*

Eppinger, S., & Browning, T. (2012). *Design structure matrix methods and applications*. MIT Press.

Estefan, J. A. (2008, May). *Survey of model-based systems engineering (mbse) methodologies* (Tech. Rep.). Pasadena, California, USA: California Institute of Technology.

Feather, M. S., Goebel, K., & Daigle, M. (2010, April). Tackling verification and validation for prognostics. In *Spaceops 2010 conference.* American Institute of Aeronautics and Astronautics. doi: 10.2514/6.2010-2183

Feather, M. S., & Markosian, L. Z. (2006). Emerging technologies for V&V of ISHM software for space exploration. In *2006 IEEE Aerospace Conference* (p. 1-15). doi: 10.1109/AERO.2006.1656133

Fitzgerald, F., Larsen, P. G., & Woodcock, J. (2013). Foundations for model-based engineering of systems of systems. In M. Aiguier, F. Boulanger, D. Krob, & C. Marchal (Eds.), *Proceedings of the fourth international conference on complex systems design and management.* Springer.

Haskins, C. (Ed.). (2010). *Systems engineering handbook: A guide for system life cycle processes and activities* (3rd ed.). International Council on Systems Engineering (INCOSE).

Honour, E. C. (2013). *Systems engineering return on investment* (Ph.D. thesis). University of South Australia.

ISO. (2003). *Condition monitoring and diagnostics of machines – Data processing, communication and presentation – Part 1: General guidelines* (Standard No. ISO 13374-1:2003). Geneva, Switzerland: International Organization for Standardization.

Jamshidi, M. (Ed.). (2008). *Systems of systems engineering*. CRC Press. doi: 10.1201/9781420065893

Kopetz, H. (2015). Simplification principles in the design of cyber-physical system-of-systems. In G. Auvray, J. C. Bocquet, E. Bonjour, & D. Krob (Eds.), *Proceedings of the sixth international conference on complex systems design and management.* Springer.

Lacaille, J. (2009). A maturation environment to develop and manage health monitoring algorithms. In *Annual conference of the prognostics and health management society 2009.*

Lacaille, J. (2010, March). Validation of health-monitoring algorithms for civil aircraft engines. In *Aerospace conference, 2010 IEEE* (p. 1-11). doi: 10.1109/AERO.2010.5446815

Lacaille, J. (2012, March). Validation environment of engine health monitoring algorithms. In *Aerospace conference, 2012 IEEE* (p. 1-11). doi: 10.1109/AERO.2012.6187369

Lamoureux, B., Massé, J. R., & Mechbal, N. (2012, June). An approach to the health monitoring of the fuel system of a turbofan. In *Prognostics and Health Management (PHM), 2012 IEEE Conference on* (p. 1-6). doi: 10.1109/ICPHM.2012.6299528

Luzeaux, D. (2013). SoS and large-scale complex systems architecting. In M. Aiguier, F. Boulanger, D. Krob, & C. Marchal (Eds.), *Proceedings of the fourth international conference on complex systems design and management.* Springer.

Markosian, L., Feather, M. S., Brinza, D., & Figueroa, F. (2005, November). V&V of ISHM software for space exploration. In *1st international forum on integrated system health engineering and management in aerospace.*

Massé, J. R., Hmad, O., & Boulet, X. (2012). System phm algorithm maturation. In *First european conference of the prognostics and health management society 2012.*

Massé, J. R., Lamoureux, B., & Boulet, X. (2011, June). Prognosis and health management in system design. In *Prognostics and Health Management (PHM), 2011 IEEE Conference on* (p. 1-5). doi: 10.1109/ICPHM.2011.6024346

OMG. (2015). *OMG Systems Modeling Language (OMG SysML ^{TM})* (Standard Specification No. formal/2015-06-03). Needham, Massachusetts, USA: Object Management Group.

Rajamani, R., Saxena, A., Kramer, F., Augustin, M., Schroeder, J. B., Goebel, K., ... Lin, W. (2013). *Developing IVHM requirements for aerospace systems* (SAE Technical Paper No. 2013-01-2333). SAE International. doi: 10.4271/2013-01-2333

Roychoudhury, I., Saxena, A., Celaya, J. R., & Goebel, K. (2013). Distilling the verification process for prognostics algorithms. In *Annual conference of the prognostics and health management society 2013.*

SAE Aerospace. (2010, December). *Guidelines for development of civil aircraft and systems* (Aerospace Recommended Practice No. ARP4754A). Warrendale, Pennsylvania, USA: Author.

Saxena, A., Roychoudhury, I., Celaya, J., Saha, B., Saha, S., & Goebel, K. (2012, June). Requirements flowdown for prognostics and health management. In *Infotechaerospace 2012.* American Institute of Aeronautics and Astronautics. doi: 10.2514/6.2012-2554

Saxena, A., Roychoudhury, I., Celaya, J., Saha, S., Saha, B., & Goebel, K. (2010, April). Requirements specifications for prognostics: An overview. In *AIAA InfotechAerospace 2010.* American Institute of Aeronautics and Astronautics. doi: 10.2514/6.2010-3398

Saxena, A., Roychoudhury, I., Wei, L., & Goebel, K. (2013, August). Towards requirements in systems engineering for aerospace IVHM design. In *AIAA In-*

fotechAerospace Conference 2013. American Institute of Aeronautics and Astronautics. doi: 10.2514/6.2013-4659

Shone, N., Shi, Q., Merabti, M., & Kifayat, K. (2011). System-of-systems monitoring: A survey. In *PGNet proceedings of the 12th annual postgraduate symposium on the convergence of telecommunications, networking and broadcasting.* Liverpool John Moores University, School of Computing and Mathematical Sciences.

The Standish Group. (1994). *CHAOS report* (Tech. Rep.). The Standish Group.

BIOGRAPHIES

Thomas Dumargue received the Diplôme d'Ingénieur from Arts et Métiers ParisTech, Paris, France and the MSc degree in Mechatronics from Loughborough University, UK in 2010. Since 2011 he has worked at Safran Aircraft Engines (formerly Snecma) in the Externals Division as a Systems Engineer working on Systems Engineering and PHM.

Jean-Robert Pougeon received the Diplôme d'Ingénieur from ISAE-ENSMA, Poitiers, France in 2002. After gaining experience in the rail transport field, he joined Safran Aircraft Engines (formerly Snecma) in 2012 to work on control systems of military aircraft engines and has been Head of the PHM team in the Externals Division since 2015.

Jean-Rémi Massé is a Senior Expert in systems dependability engineering and health monitoring at Safran Aircraft Engines (formerly Snecma). He has founded the dependability engineering department of Safran Aircraft Engines. He has a Ph.D. in statistics and has practised and taught statistics in several companies and universities.