Agile Systems Engineering: an Iterative and Collaborative Approach for Complex Systems Development

Sevag AINEJIAN[1], Guy-André BOY[2], Nicolas CHARLIER[3], Yann DECRE[4], Daniel KROB[5], and Loic LE SAUCE[6]

[1] CESAMES – 10, rue de Penthièvre – 75008 Paris – France – email: sevag.ainejian@cesames.net
[2] CentraleSupélec – France – email: guy-andre.boy@centralesupelec.fr
[3] CESAMES – 10, rue de Penthièvre – 75008 Paris – France – email: nicolas.charlier@cesames.net
[4] CESAMES – 10, rue de Penthièvre – 75008 Paris – France – email: yann.decre@cesames.net
[5] CESAMES – 10, rue de Penthièvre – 75008 Paris – France – email: daniel.krob@cesames.net – Corresponding author
[6] CESAMES – 10, rue de Penthièvre – 75008 Paris – France – email: loic.lesauce@cesames.net

Abstract

This paper presents an **agile systems engineering framework** especially dedicated for developing complex systems, that is to say mixed hardware and software systems where physics is critical. Our approach consists in seeing an agile complex system development project as a mechanism, controlled by the key design drivers associated with a system, for quickly refining its design space. It was validated in practice on several aeronautic development projects and showed gains, compared to projects of similar size and complexity, of the order of – 30 % on design schedules and of the order of – 50 % on resource workload and costs.

Introduction

Agile development methods are nowadays well established in **software engineering**. They refer to an approach to software development under which requirements and solutions evolve simultaneously through the collaborative effort of self-organizing cross-functional teams, including customer(s) and end-users(s), whose objective is to construct a software solution that answers as well as possible to its business needs. Agile approaches rely on adaptive planning, evolutionary development, early delivery, continuous improvement and rapid & flexible responses to changes (see [13] or [32]).

The term *agile* was popularized in the software area by the *Manifesto for Agile Software Development* that goes back to 2001 [1]. The values and principles proposed by this manifesto are the cornerstones of a broad range of modern agile software development frameworks, including Scrum and Kanban, among the most popular agile methodologies in these contexts [9]. Such agile approaches are used nowadays widely in the industry for non-critical software development: most service industries (e.g., banks, insurance, retail, telecommunications, etc.) are typically using them daily for developing their business software capabilities. All this is also possible when a human systems integration is explicitly considered, that is, when an appropriate human-in-the-loop modeling and simulation approach is used to enable incremental discovery of emerging behaviors and properties, leading to emergent functions and structures of the overall sociotechnical system [6].

The maturity of agile approaches in software development can be measured by the worldwide success of **agile frameworks at enterprise scale**, such as SAFe®, standing for Scaled Agile Framework (see [18], [19], [33] or [34]). This last framework proposes an entire body of knowledge for deploying and scaling agile methods in a enterprise: SAFe® indeed explains how to use consistently agile approaches both at team, program, large solutions, and enterprise portfolio levels, again still in a software context.

However, the same maturity does not exist for **complex systems**, meaning here the development of mixed hardware and software systems where physics remains critical, as this notion is typically addressed in systems engineering (see [2], [3], [4], [5], [8], [14], [15], [16], [20], [21], [22], [24], [25], [26], [29], [35], [36], [37], [38] or [39]), where one can only find some poor and still relatively recent references to *agile systems engineering*. The first attempt of extending the agile framework to the context of system development seems to only go back to the end of 2012 when an IBM researcher, Hazel Woodcock, proposed a revisit of the Agile Manifesto for systems engineering [40]. In line with this seminal initiative, a working group of the International Council on Systems Engineering (INCOSE) started then in 2014 to work on agile systems engineering [17], leading in particular to a first textbook published by B.P. Douglass at the end 2015 [11]. Finally, one shall also point out a recent attempt – that goes back to October 2017 – of SAFe® team that proposed a sketch of a Model-Based Systems Engineering agile framework. This last proposal remains, however, quite poor and is not supported by returns on experiments from real system developments [34].

Moreover, these attempts to extend the agile paradigm to systems engineering did not deeply penetrate the industry. In practice, most industrial development organizations are indeed not "agile" at all and are still using quite classical, more or less cascaded development approaches based on a V-cycle organization with specialized teams working in silos, where customer-focus and transversal collaboration are usually weak. This situation is probably because physical systems are much more difficult to handle than software systems, and systems development frameworks are usually strongly constrained by regulatory issues, especially when dealing with critical systems.

Nevertheless, some relatively rare industrial companies have successfully started to deploy agile systems engineering approaches with outstanding outcomes. Two examples are SpaceX and Gripen, which are presented below.

- First of all, SpaceX has been able to apply an "agile" approach to a multidisciplinary industrial and complex system such as a launcher. Agility allowed SpaceX to significantly reduce program costs by transforming the classical unique single development cycle into a series of short design-build-test development cycles. During these short cycles, departments and disciplines are aligned on the same system view of the global product. All development actors involved in SpaceX engineering activities are regularly synchronized to reduce over-costs and delays in each short iterative cycle. In the same way, customers' requirements are permanently tracked and verified to optimize all design items covering these requirements at each iterative design-build-test cycle. Finally, it should be noted that digital continuity is critical to improving collaboration and controlling the state of the system at each cycle, allowing the three SpaceX sites, each involving numerous specialized teams, to work as a global team sharing all the critical data. However, no detailed information does exist on the SpaceX case, and it is thus impossible to derive a specific agile methodology from it.
- In the same way, Saab successfully applied an iterative approach to developing the JAS 39 E/F Gripen military aircraft, with a considerable impact on design complexity. This aircraft's design cost is estimated at 14 B€ instead of 25 B€ for the Dassault Aviation's Rafale. The Gripen's success is mainly linked to the efficiency of the transversal collaboration between functional and design teams, who used digital and functional mockup tools as pivot tools to synchronize, which is critical but is, however, just a part of an agile approach.

These two examples show well the strategic importance of agility in the context of multi-physical complex industrial system development, though not leading to any explicit methodology.

Our Agile Systems Engineering Framework

The first fundamental of the agile systems engineering framework that we are proposing here consists in analyzing an agile complex system development project as a *mechanism for quickly refining the design space associated with the system under development* that allows it to move:

- From a starting point or an initial situation where the target system can only be located in a more or less large area of its design space,
- To a point of arrival or to a final situation where the target system is now reduced to a well - specified point in its design space.

This approach is thus based on the classic concept of *design space* of a system, introduced in systems engineering by MIT (see [31],[18], [28] or [12]) and that we shall now present more in details.

The Concept of Design Space

The classic definition of a design space (see [31], [18], [28] or [12]) states: "*The design space of a system is the whole set of all the conceptions that meet the needs of the stakeholders of this system. From a mathematical point of view, it is a topological space. The problem of designers is to find the most optimal design – in the sense of Pareto – within this space*".

To fully understand this fundamental concept, one must remember that a given system can be seen as a multi-dimensional object whose dimensions – in an algebraic meaning – are the independent variables that allow it to be specified. To fix the ideas, imagine that we want to design a colorful cube C which depends on three design parameters, namely:

- The length of a side of C, which is a real positive number, varying in R+,
- The color of C, which varies in a discrete set Colors of colors,
- The mass of C that can also be seen as a real positive number, varying in R+.

We can then associate with our cube a three-dimensional space E(C) = R+ x Colors x R+, which can be equipped with a quotient topology, in order to capture the fact that mass and length are here not independent since these two parameters are coupled as soon as we have frozen the material that we use to make a cube, where each of the components of our topological space has its natural topology (here continuous, discrete and continuous).

The design space associated with cube C is then defined as the part of the quotient topological space E(C) corresponding to all the cubes achievable in practice in a given context.

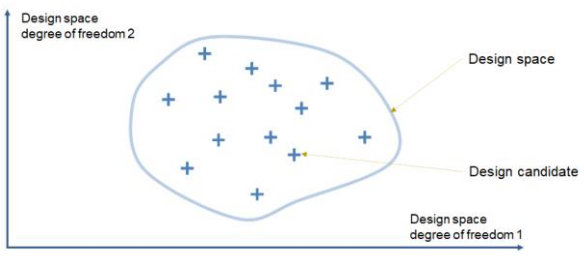

Figure 1 : A design space with two degrees of freedom

This example is straightforward, but it is easy to understand how to generalize it. In practice, the design spaces of complex systems are parts of very large topological spaces (from hundred thousand to several million dimensions, depending on the system's complexity), the number of dimensions in play corresponding to the maximum number of independent design parameters that characterize a given system. It is thus a complex mathematical object, both interesting to consider conceptually and not without practical interest because many engineering problems can be reduced to multi-criteria optimization issues – that we can approach with Pareto front calculation methods – within the design space associated with a given system (see [5], [7], [8] or [10]) .

Figure 1 finally shows a hyper-simplified example of a two-dimensional design space which we will use to illustrate this concept. Each conceivable system corresponds here to a point of the design space which is the space surrounded by the border curve in this figure and delimiting the design perimeter of all conceivable systems.

Agile Systems Engineering as a Design Space Exploration Process

That being recalled, a complex system design process can be seen as a process of exploring the design space of the considered system, whose role is to identify the target solution by analysing alternative solutions, then evaluating them with regard to selected performance criteria to eliminate non-optimal solutions and keeping only relevant solutions before exploring them further.

In the same way, an *agile systems engineering process* can then be seen as a design space exploration process whose aim is to identify, as quickly as possible, the viable design perimeter by considering, analysing and comparing, at the same time, several design hypotheses at different levels of granularity, as represented in Figure 2 where each bold dashed circle or point represent a design alternative.

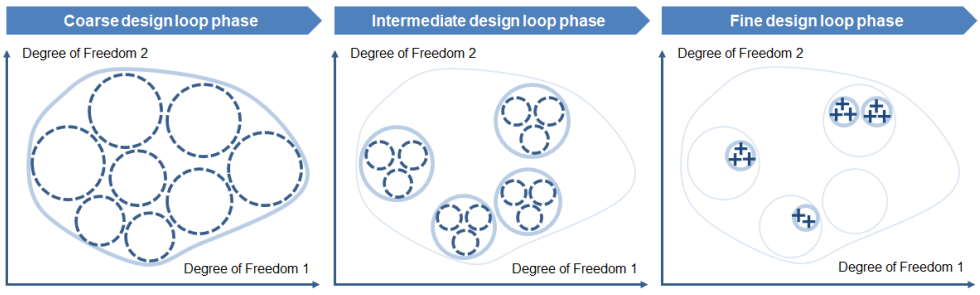

Figure 2 : Principle for exploring the design space in an agile way

In this approach, the design is organized in design loops per stages of refinement (coarse design loop, intermediate design loop, fine design loop) dedicated to more and more fine levels of granularity. To take an automotive example, the first step (coarse design loop) would correspond to the choice of the global dimensions of a vehicle (e.g., space, power, length, etc.), the second step (intermediate design loop) to the dimensioning of large subsystems of a vehicle (e.g., engine, chassis, bodywork, etc.) and the third step (fine design loop) to the detailed specifications of modules or parts of a vehicle.

Once the concept of design space has been identified as a central concept for “thinking” agile systems engineering, one still must know how to handle it in practice, a design space being a terribly abstract topological space. For this, we had to define a mean of concretely representing and dealing with the design space of a complex system, which allows both to have a good level of coverage, while remaining practical and actionable in practice along real industrial projects.

To do this, we propose the concept of *key design driver*:

"*Key Design Drivers (KDD) are system design parameters of capital importance for design. The values of the Key Design Drivers are produced by design activities in order to assess the design alternatives and to arbitrate between them. They shall be defined in the form of a variation range*".

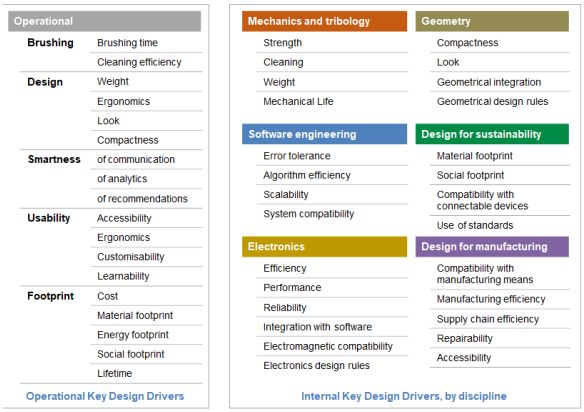

Figure 3 : Example of key design drivers for an electronic toothbrush system

For each design loop, the key design drivers must be a coherent set of interdependent characteristics covering both the operational architecture of the product (e.g., mission profiles, use cases, stakeholder needs), especially including manufacturing and maintenance constraints, its functional architecture (e.g., specific / critical functions, behavioral performance) and its structural architecture (e.g., mass, geometry of critical parts, technological requirements, etc.).

Key design drivers can typically be identified by eliciting the needs and the functional and structural requirements of a given system according to the CESAM methodology [23] and then extracting the key performance indicators that are involved in these system properties (see Figure 3 for an example on an electronic toothbrush system). One must finally eliminate the key performance indicators that depend on others to get the key design drivers in our meaning. The system design space becomes then a part of a topological space with N dimensions, where N is the number of key design drivers.

Main Features of our Agile Systems Engineering Framework

We can now present the main features of an **agile model-based systems engineering framework**, based on the fundamentals presented above for its agile part and on the CESAM framework for its model-based systems architecting part [23], which emerged from concrete industrial experiences in order to ensure practical applicability (see the last section).

Feature 1: Product / Organisation Alignement

Preliminary definition of a product reference architecture:

Adequate product architecture is the first key element to enable agile systems engineering. In this matter, it is vital to construct a generic reference product breakdown where the system of interest is recursively decomposed into logical components that are as independent as possible from a functional and logical point of view. Such a breakdown helps reduce the functional and logical interfaces between the system's components and, consequently, the issues linked to the concurrent design and integration of these components. Such a generic reference product breakdown shall, in particular, contribute to a

maximalist ("150 %") high-level vision of the system or the family of systems of interest that includes the definition of generic assets such as generic external and internal interfaces, generic needs, and functional and structural requirements, generic functional and logical architectures, etc.

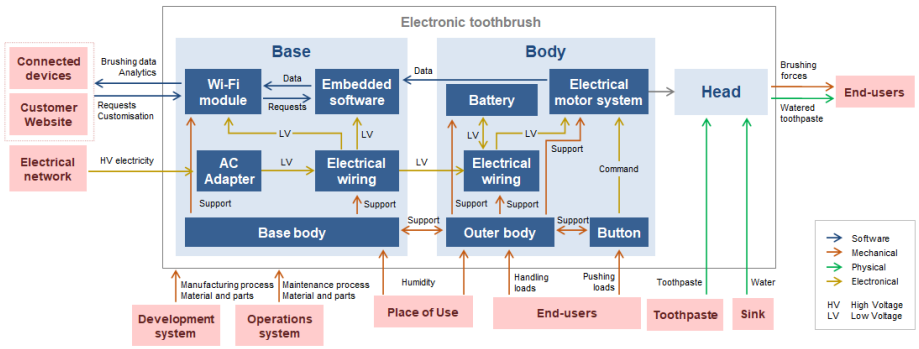

Figure 4 – Reference logical architecture of an electronic toothbrush

An example of such a generic reference architecture – here a logical one – is provided in Figure 4 for an electronic toothbrush. Note that these reference architectures are key for providing a standard and shared way of presenting and analyzing different design alternatives when relevant. They are also a natural support for impact analyses, as far as they have a double functional and structural nature.

Product / organisation alignment:

A second critical element of agile systems engineering is **aligning the multidisciplinary activities and the generic reference product architecture**. Organizations shall replicate the product architecture to provide the best agility throughout the development process. Traditional activity organization is often a source of tunnel effect since teams are not functionally independent, not sufficiently autonomous, and even sometimes too big to be mastered in an agile process. It is thus often crucial to reorganize the development teams to mimic the reference product architecture in the organization.

Hence, a typical agile system development organization shall be split into 2-3 or more layers according to product complexity and constraints. For instance, a typical 3 layers organization will be broken down into a system level, a sub-system level, and a part level (this last level considers the part manufacturing constraints, typically for mechanical components). In this organization, each team shall be in charge of a unique component of the reference breakdown of the system of interest, all components being covered by the different development teams. Finally, one must identify “architects” that shall be in charge of synchronizing the various development teams horizontally between the different disciplines and vertically between the different product layers. The resulting organizational model is illustrated in Figure 5 for an electronic toothbrush.

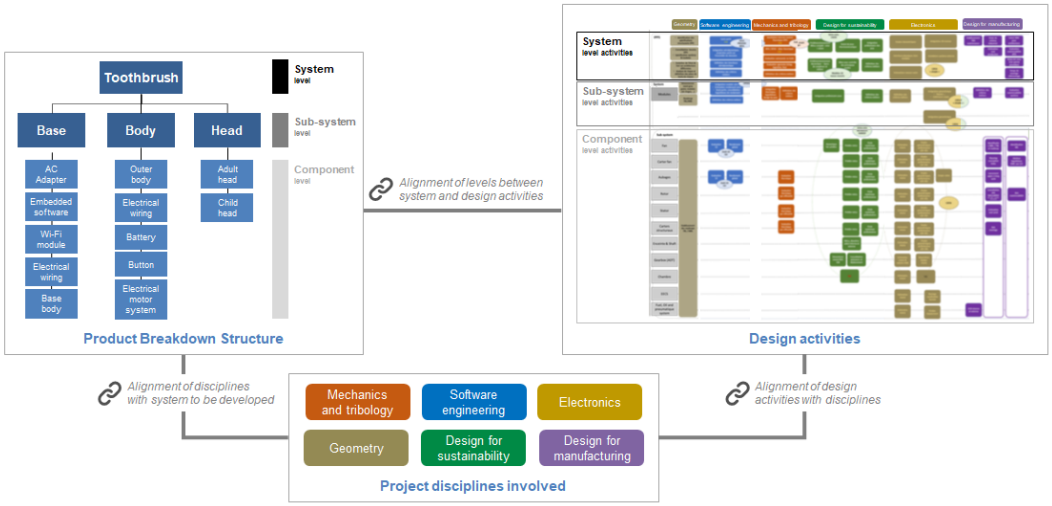

Figure 5 – Agile organization resulting from a product / organization alignment

Feature 2: Iterative Collaboration between Layers and Disciplines

The second idea of our agile approach for complex systems engineering consists in **iteratively defining the system** by organizing its development into **successive collaborative design loops** of increasing length and granularity where the system relevancy, consistency, and feasibility are assessed at more and more precise levels of analysis (see Figure 6). Each loop shall cover and study several possible system variants at different grain levels along their full development cycle, starting from top-level architecture up to manufactured part design and definition of manufacturing & maintenance concepts. The design choices shall be gradually frozen and deepened until the system's full specification and associated manufacturing and maintenance processes are achieved. All system stakeholders – in particular manufacturing and maintenance actors – shall be involved in these design loops to capture their constraints as early as possible.

Such a development model makes it possible to freeze quickly structuring choices related to system global performance, system architecture, external or internal interfaces, critical parts or technologies and development, manufacturing and maintenance processes, ensuring the technical and industrial feasibility of the target system and providing a good control of risks at any time.

During each design loop, all involved teams – that is, **all engineering and manufacturing disciplines** involved in a product development – shall work collaboratively and share the same set of key design drivers and hypotheses that will be progressively refined and frozen (see feature 3). At the end of each design loop, a new system increment shall be generated and validated through virtual integration based on relevant models and tools (see feature 4). Due to the progressive refinement of the key design drivers, this approach provides considerable adaptability of the system development process to new or updated requirements or constraints from various stakeholders and helps multidisciplinary teams reaching the global optimum system choice at each loop. Figure 6 illustrates it with four design loops starting from an initial high-level coarse-grain loop up to a final fine-grain detailed loop.

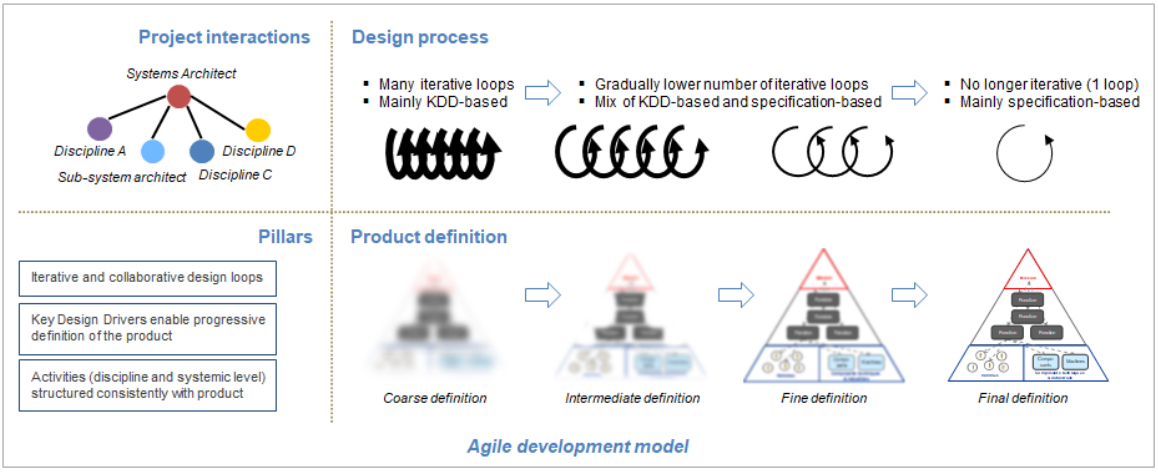

Figure 6 – Our agile development model based on successive and more & more detailed design loops

Each design loop will, in particular, analyze several design alternatives, starting typically with a dozen options in the first loop, reducing them to 2-4 main choices in the second loop, up to converging to only 1-2 critical alternatives in the last loop. This can be achieved with trade-off techniques (cf. [27] or [30]), with different levels of details to reach a global optimum.

Note that system requirements are usually often over-specified and in much too significant numbers since all actors in the development chain often create requirements to protect themselves by taking local safety margins. In an agile approach, each design loop shall pay specific attention to formalize the system requirements to **what is necessary**. This objective mainly relies on classic prioritization / multi-criteria optimization techniques managed through collaborative requirements reviews, regularly involving all key development actors where performance allocations are collaboratively managed and optimized when descending into product architecture.

Since we are dealing with systems, any agile development process shall also consider deeply the constraints coming from manufacturing and maintenance since these are the steps where the “real thing” is constructed and used. In an agile systems-engineering process, manufacturing teams must be especially involved in each design loop – as already pointed out – to analyze design requirements from their perspective and to evaluate / simulate their impact on manufacturing and assembly processes with the “good” industrial performance indicators (e.g., lead time, non-recurring and recurring costs, non-quality). Similarly support and services teams must be involved to evaluate the impact of design requirements on the maintenance and support processes using their relevant performance indicators (e.g., time and cost of maintenance and support operations).

Regular systems architecture rituals, involving design, manufacturing and maintenance teams, and more generally all relevant stakeholders, shall be regularly organized within each design loop to integrate all these different point of views in the design. Model-based systems engineering plays a key role in these rituals which shall be balanced between a product and a project focus. A typical agile design ritual indeed consists in examining and discussing collectively more and more precise models of the product under development and not only the status of the project activities. These rituals rely naturally on a set of operational, functional and structural system models [23] whose discussion, challenge and convergence shall be at their heart.

Feature 3: Iterative and Shared Key Design Driver's Management

The third key idea is the **use of the key design drivers**, as introduced previously, to monitor each design loop. They indeed allow both to explore the design space in a rigorous and controlled way, but also to identify potential architecting variants (see Figure 7).

The key design drivers shall especially represent the hypotheses shared between all teams involved in agile development, i.e., elicit a shared design space. The first activity is identifying the key design drivers associated with the system of interest. The first loop shall focus then on the structuring parameters of the system that are to be analyzed with the highest priority. During a given design loop, teams shall work with a shared fixed range for each key design driver, with freedom of design. At the same time, their parameters are contained in the defined ranges associated with each key design driver from a given synchronization ritual to the next one. At each iteration, the teams shall define and evaluate potential architecture variants within the ranges of the considered key design drivers to reduce these ranges at the end of the iteration. If no feasible architecture can be defined within the design space defined by the current key design drivers, a synchronization activity will redefine new ranges for the key design drivers.

This approach ensures agility to adapt the design in a robust way by mastering the traceability from initial stakeholder requirements to shared design functional and structural hypotheses.

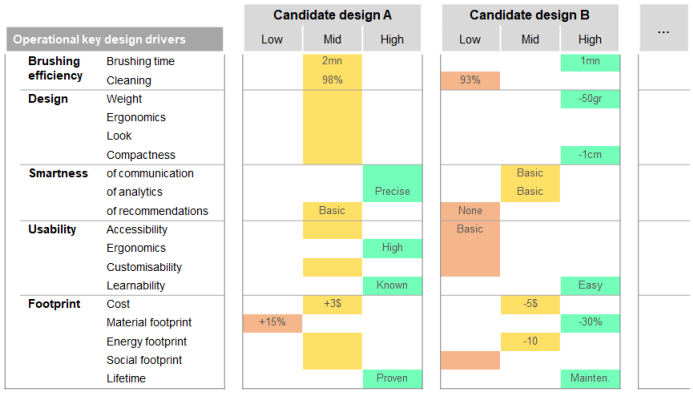

Figure 7 – Examples of key design drivers identified in the context of an electronic toothbrush development, here used to define two product variants that were analysed during a first design loop

Feature 4: DMU and FMU as Pivots for Transversal Synchronization

The last ingredient of an agile systems engineering process is to **use digital and functional mockups (DMU and FMU) as pivots for virtual integration and transversal synchronization** in each design loop. The key idea is here that at the end of each iteration, each team shall produce models (3D, thermal, mechanical resistance, thermodynamics, etc.) which are to be integrated into a shared DMU (for the geometrical aspects) and a shared FMU (for the functional aspects), whose overall coherence is analyzed. Due to the need to synchronize all disciplines at each product increment, using these digital models as pivots for the design is crucial to supporting an agile approach. In this matter, note that the functional digital mockup is usually not a single tool but a series of interconnected tools (e.g., a system modeling tool, a simulation workshop, a dysfunctional analysis tool, etc.).

A key difficulty is primarily here to ensure the alignment of the system breakdown of the functional digital mockup with the geometrical breakdown of the digital mockup, which currently is still a matter of human expertise. Both tools have different modeling constraints that need to be addressed in an integrated way. Nevertheless, functional and discipline integration through functional-oriented and computer-aided design tools is crucial in implementing agile systems engineering.

The critical point is that the functional mockup shall be designed to provide comprehensive system-level predictive models, i.e., integrating all key product dimensions, typically multi-physical, that allow simulating the system under design within its environment to validate hypotheses and manage trade-offs. The levels of detail and the representativeness of such models must be aligned with the granularity of the design loops. These models can be existing models (e.g., Simulink or Modelica models) or existing model aggregations via ad hoc tools (e.g., Optimus). Such system-level simulation models are key for evaluating design performances, validating the design choices, managing the initial trade-offs, and helping to negotiate with the stakeholders.

Discussion

Our agile systems engineering framework has the following four key features, allowing an industrial company to successfully implement an agile/iterative systems engineering framework:

1. Product and organization alignment,
2. Iterative collaboration between layers and disciplines,
3. Iterative key design driver's management,
4. Digital & functional mockups as pivots for transversal synchronization.

Note that features 1 and 3 do strongly rely on systems engineering and architecting techniques, when feature 2 is rather an adaptation of agile and SAFe® principles to industrial contexts and feature 4 refers to the digital tooling support. The synthesis of all these features is illustrated in Figure 8.

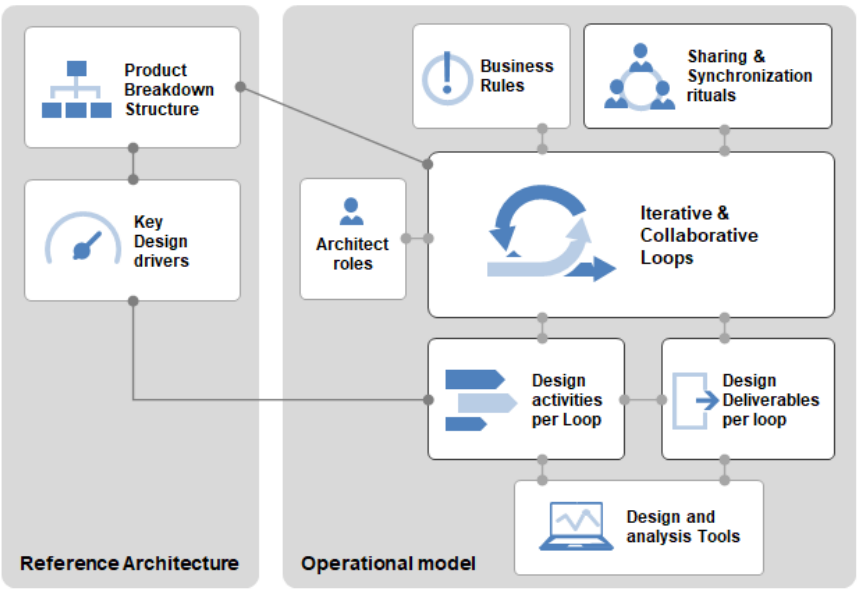

Figure 8 – Overview of our agile systems engineering framework

Experimental Validation

Our agile systems engineering framework was progressively constructed through a series of concrete experiments in three different types of industrial environment, namely aeronautic, nuclear plant and civil engineering contexts to cover different system scales.

We then used again an aeronautic context to test our complete methodology on various real systems. To this aim, the complete agile model-based systems engineering framework was then experimentally deployed and validated in an industrial context provided by an international aeronautical player on four varying-scale innovation projects involving complex systems with a dominant mechanical focus (ranging from complex parts to a complete integrated system). It should be noted that the deployment of this agile approach was part of an enterprise transformation plan with the objectives of accelerating development and better exploring the design spaces, the projects serving as "proofs of concept."

The methodology for deploying the agile approach on projects followed a similar four-step pattern, based on the principles that we presented, and strongly involved project key actors:

- Step 1: **definition of the system architecture** of the system of interest, through operational, functional and structural views, supplemented, when necessary, by relevant manufacturing or assembly diagrams;
- Step 2: **identification of the key design drivers**, associated with all disciplines involved in the design of the system of interest;
- Step 3**: specification of the design loops** – according to the planning of the project of interest – by characterizing in particular the objectives & duration of each loop, the key design drivers managed during each loop, the design activities (modeling, analyses, simulations, etc.) of each loop and the engineering data flows feeding these design activities;
- Step 4: **deployment of agile operation and loops**, as defined in step 3, according to several iterations, taking into account the feedbacks at each loop.

The deployment of our agile framework on projects especially revealed the following elements:

1 The agile mode of operation obliges formal synchronization of design activities which was usually managed in an uncontrolled way.

2 The implementation of design loops – especially during the upstream design phases – leads to challenge the level of detail of design activities, allowing to better explore the design space.

3 The digital continuity of design tools, in terms of data configuration management and data exchange, as well as the collaborative management of key design drives is a key success factor.

4 Classical agile principles such as collaborative work, co-location of teams and visual management are also key success factors.

Finally, the performance measured during the four innovation projects on which we deployed our agile systems engineering framework on the upstream phases of the critical design review showed gains, compared to projects of similar size and complexity, of the order of – 30 % on design schedules and of the order of – 50 % on resource workload and costs.

References

[1] Agile Manifesto, *Agile Manifesto*, http://agilemanifesto.org/ (2001)

[2] ANSI/GEIA, *ANSI/GEIA EIA-632 – Processes for engineering a system* (2003)

[3] Aslaksen E.W., *The changing nature of engineering*, McGraw-Hill (1996)

[4] Aslaksen E., Belcher R., *Systems engineering*, Prentice Hall (1992)

[5] Blanchard B.S., Fabricky W.J., *Systems engineering and analysis*, Prentice Hall (1998)

[6] Boy G.A., *Human Systems Integration: From Virtual to Tangible*. CRC Press, Taylor & Francis Group (2020)

[7] de Weck O., *Strategic Engineering – Designing systems for an uncertain future*, MIT (2006)

[8] de Weck O.L., Roos D., Magee C.L., *Engineering systems – Meeting human needs in a complex technological world*, The MIT Press (2011)

[9] Dingsoyr T., Nerur S., Balijepally V., Moe N.B., *A decade of agile methodologies: Towards explaining agile software development*, Journal of Systems and Software, 85, (6), 1213-1221 (2012)

[10] Doufène A., Krob D., *Pareto Optimality and Nash Equilibrium for Building Stable Systems*, IEEE International Systems Conference (2015)

[11] Douglas B.P., *Agile Systems Engineering*, Morgan Kaufman (2015)

[12] Hegedüs, Á., Horváth, Á., Varró, D. *A model-driven framework for guided design space exploratio*n. *Autom Softw Eng* **22**, 399–436 (2015). doi.org/10.1007/s10515-014-0163-1

[13] Highsmith J., *Agile Project Management: Creating Innovative Products,* Addison-Wesley (2009)

[14] Honour E.C., *Understanding the value of systems engineering*, INCOSE 2014 International Symposium, Vol. 14, 1207–1222, Toulouse, June 20-24, 2014, France, INCOSE (2014)

[15] IEEE, IEEE 1220-2005 – *Standard for Application and Management of the Systems Engineering Process*, Institute of Electrical and Electronics Engineers (2005)

[16] INCOSE, *Systems Engineering Handbook, A guide for system life cycle processes and activities*, INCOSE (2011)

[17] INCOSE, *Agile systems*, https://www.incose.org/ChaptersGroups/WorkingGroups/transformational/agile-systems-se

[18] Kang E., Jackson E., Schulte W., *An Approach for Effective Design Space Exploration*, [in "Foundations of Computer Software. Modeling, Development, and Verification of Adaptive Systems", Calinescu R., Jackson E., Eds.], Monterey Workshop 2010, Lecture Notes in Computer Science, 6662. Springer (2011). doi.org/10.1007/978-3-642-21292-5_3

[19] Knaster R., Leffingweel D., *SAFe® Distilled: Applying the Scaled Agile Framework for Lean Software and Systems*, Addison-Wesley (2017)

[20] Kossiakoff A., Sweet W.N., *Systems engineering – Principles and practice*, Wiley (2003)

[21] Krob D., *Eléments d'architecture des systèmes complexes*, [in "Gestion de la complexité et de l'information dans les grands systèmes critiques", A. Appriou, Ed.], 179-207, CNRS Editions (2009)

[22] Krob D., *Eléments de systémique – Architecture de systèmes*, [in "Complexité-Simplexité", A. Berthoz - J.L. Petit, Eds.], Editions Odile Jacob (2012)

[23] Krob D., *Model-Based Systems Architecting – Using CESAM to Architect Complex Systems*, ISTE-Wiley (2022)

[24] Maier M.W., Rechtin E., *The art of systems architecting*, CRC Press (2002)

[25] Meinadier J.P., *Ingénierie et intégration de systèmes*, Lavoisier (1998)

[26] Meinadier J.P., *Le métier d'intégration de systèmes*, Lavoisier (2002)

[27] Miles L.D., *Techniques of value analysis and engineering*, McGraw-Hill (1972)

[28] Nardi L., Koeplinger D., Olukotun K., *Practical Design Space Exploration*, [in "2019 IEEE 27th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems"], 347-358, (2019). doi:10.1109/MASCOTS.2019.00045

[29] NASA*, Systems Engineering Handbook,* 2007-edition, NASA/SP-2007-6105 (2007)

[30] Parnell G.S., Driscoll P.J., Henderson D.L., *Decision marking in systems engineering and management*, Wiley (2011)

[31] Pimentel A.D., Explor*ing Exploration: A Tutorial Introduction to Embedded Systems Design Space Exploration*, [in "IEEE Design & Test"], 34, (1), 77-90, (2017). doi: 10.1109/MDAT.2016.2626445

[32] Runyan K., Ashmore S., *Introduction to Agile Methods*, Addison-Wesley (2014)

[33] SAFe, *Scaled Agile Framework*, http://www.scaledagileframework.com/

[34] SAFe, *Model Based Systems Engineering*, http://www.scaledagileframework.com/model-based-systems-engineering/

[35] Sage A.P., Armstrong J.E., *Introduction to systems engineering*, Wiley (2000)

[36] Sillitto H., *Architecting systems – Concepts, principles and practice*, College Publications (2014)

[37] Simon H., *The Architecture of Complexity*, Proceedings of the American Philosophica, 106 (6), 467-482 (1962)

[38] Turner W.C., Mize J.H., Case K.H., Nazemetz J.W., *Introduction to industrial and systems engineering*, Prentice Hall (1978)

[39] von Bertalanffy K.L., *General System Theory: Foundations, Development, Applications*, George Braziller (1976)

[40] Woodcock H., *The Agile Manifesto reworked for Systems Engineering*, INCOSE UK, ASEC conf. (2012)