CESAF: an Iterative & Collaborative Approach for Complex Systems Development

Daniel KROB[1]

1. 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 advocate adaptive planning, evolutionary development, early delivery and continuous improvement. They also encourage rapid and flexible responses to changes (see for instance [34], [56] or [68] for more details).

The term *agile* was popularized – in the software engineering area – by the *Manifesto for Agile Software Development* that goes back to 2001 (cf. [1]). The values and principles proposed by this manifesto are especially the cornerstones of a broad range of modern software development frameworks, including Scrum and Kanban, which are among the most popular agile methodologies in these contexts (see [24] for a short history of agile methodologies). One shall finally point out that such agile approaches are nowadays widely used in the industry for non-critical software development: most of service industries (banks, insurances, retail, telecommunications, etc.) are typically using them on a daily basis for developing their business software capabilities.

The maturity of agile approaches in software development can be measured by the worldwide success and dissemination of **agile frameworks at enterprise scale**, such as SAFe® (whose acronym stands for Scaled Agile Framework) which is the most famous one in this matter (see [40], [57] or [58]). This last framework proposes a full body of knowledge for deploying and scaling agile methods at enterprise level: SAFe® indeed explains how to use consistently agile approaches both at team, program, large solutions and enterprise portfolio levels, again still in the context of software development.

The same maturity does however not exist in the context of **complex systems development** where one can only find some poor and still quite recent references to *agile systems engineering*. The first attempt of extending the agile framework to the context of system development seems for instance 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 (see [76]). In the line of this seminal initiative, a working group of the International Council on Systems Engineering (INCOSE) started then – in 2014 – to work on agile systems engineering (see [38]) with a regular production on this subject, leading in particular to a first textbook which was published by B.P. Douglass in end 2015 (see [28]). Finally, one shall also point out a quite recent first attempt – that goes back to October 2017, as far as we could trace it – of SAFe® team that proposed a sketch of a Model-Based Systems Engineering agile framework. This last proposal is however reduced to a very small number of ideas, not detailed at all and clearly not very effective and not supported by returns on experiments from real system developments (see [58]).

[1] CESAMES – 15, rue La Fayette – 75009 Paris – France – email : daniel.krob@cesames.net

Moreover, these attempts for extending the agile paradigm to systems engineering clearly did not deeply penetrate the industry. In practice, most of the industrial development organizations are indeed not "agile" at all – taking here this term in the meaning of agile methods – 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. One may observe that this situation is probably due to the fact that physical systems are much more difficult to handle than software systems and that systems development frameworks are usually strongly constrained by regulatory issues, especially when dealing with critical systems.

Nevertheless, we can however see that some industrial companies have successfully started to deploy agile systems engineering (also called iterative systems engineering) approaches with outstanding outcomes. Two of these examples are SpaceX and Gripen that are quickly presented below.

- First of all, SpaceX was for instance 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 also regularly synchronized to reduce over-costs and delays on each short iterative cycle. In the same way, customers' requirements are permanently tracked and verified in order to optimize all design items covering these requirements at each iterative design-build-test cycle.Note finally that digital continuity is key to improve collaboration and to master dashboarding status at each cycle, allowing the three SpaceX locations, each of them mixing many teams and sub-teams, to work as a global team sharing all key information.
- In the same way, Saab applied successfully an iterative systems engineering approach for developing the JAS 39 E/F Gripen military aircraft, with a huge impact on design complexity. The design cost of this aircraft is indeed estimated at 14 B€ instead of 25 B€ for the Dassault Aviation Rafale. The Gripen success is considered as mainly linked to the efficiency of the transversal collaboration between functional & design teams, who used digital and functional mockup tools as pivot to synchronize.

These two examples are showing well the strategic importance of agility in the context of multi-physical complex industrial system development.

2. CESAF: CESAM Engineering Systems Agile Framework

2.1 History

CESAF [2] is an **agile model-based systems engineering framework**, based both on real concrete applications & experiences in order to ensure practical applicability and on the CESAM model-based systems architecting framework (see [17] for more details).

We already have defined the fundamentals of such a framework which, even if not fully formalized, was already used in several real development contexts in order to find quickly an optimal system that answers to all stakeholders' needs. More specifically, we helped implementing since around three years several agile development loops in complex system development contexts such as automotive

[2] CESAF is an acronym for CESAM Engineering Systems Agile Framework.

(e.g. development of new advanced driver assistance services), high tech (e.g. development of a new camera in Japan and of a smart site management system in China), energy (e.g. development of a smart energy distribution system) and aeronautics (e.g. mechanical engine design and manufacturing). In aerospace, the development of a new aircraft turbomachine was re-analyzed last year through a complete coarse grain agile loop achieved in 3 months: it allowed fixing the main known engineering and manufacturing issues through the understanding of the key relationships between product features and characteristics of the most important manufactured parts. A second experience at engine level confirmed the achievement of – 20 % of lead time, as experienced with the first project with the same customer.

Finally, we are working – since mid 2018 – within one of the key aircraft manufacturers in order to implement an ambitious transformation program at enterprise level, dedicated to the deployment through engineering, manufacturing and customer services of an agile development framework relying on model-based systems engineering. Note that these different experiences are of course confidential since they were all deployed on real development projects.

2.2 Key Features

2.2.1 Feature 1: Product / Organisation Alignement

Preliminary definition of a product reference architecture:

The first key element to enable agile engineering is an **adequate product architecture**. In this matter, it is key to construct first a generic reference product decomposition where the product is recursively decomposed into logical components which are as independent as possible from a functional & logical point of view. Such a decomposition will help reducing the functional & logical interfaces that are existing between product components, and consequently the issues linked to the concurrent design and the integration of these components. Such a generic reference product decomposition shall in particular contribute to a maximalist ("150 %") high-level vision of the product of interest (or of a family of similar products if one is dealing with a product line approach) that include the definition of generic assets such as generic external & internal interfaces, generic needs and functional & constructional requirements, generic functional and logical architectures, etc.

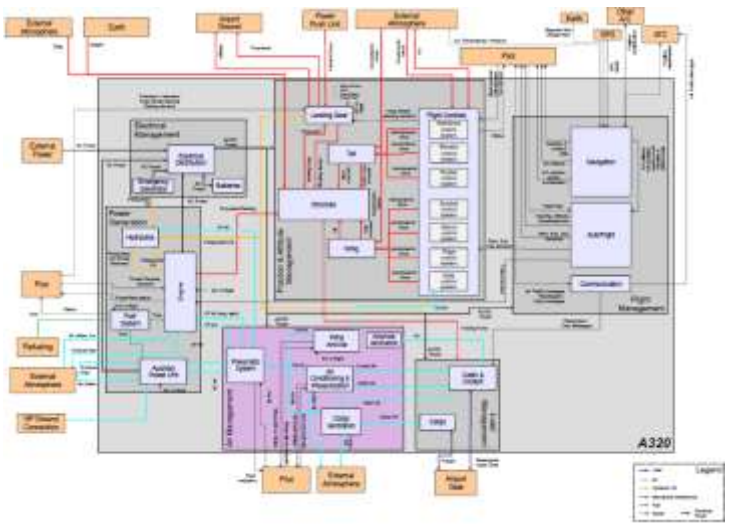

Figure 1 – Reference functional-logical architecture of a single-aisle aircraft (Airbus A320)

An example of such a generic reference architecture – that we constructed on the basis of publicly available documentation – is provided in Figure 1 for an Airbus single aisle aircraft. Note that these reference architectures are key for providing a standard & 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 constructional nature.

Product / organisation alignment:

Based on the previous reference product architecture, a second key element towards agile engineering is the **alignment of the multidisciplinary activities and of the generic reference product architecture**. Organization shall indeed replicate the product architecture in order to provide the best agility all over the development process. Traditional activity organization is indeed often a source of tunnel effect due to the fact that teams are not functionally independent, not sufficiently autonomous and even sometimes too big to be mastered in an agile process. It is therefore often key to reorganize the development teams in order to mimic the reference product architecture in the organization.

As a consequence, a typical agile development organization shall be split into 2 - 3 or more layers according to product complexity and constraints. A typical 3 layers organization will for instance be decomposed into a system level, a sub-system level and a part level (this last one taking into account the part manufacturing constraints, typically for mechanical components). In this organization, each development team shall therefore be in charge of a unique component of the reference product decomposition of the product of interest, all components being covered by the different development teams. Note finally that one also has to identify "architects" that shall be in charge of synchronizing the development teams, both horizontally between the different disciplines and vertically between the different product layers. The resulting organizational model is illustrated in Figure 2.

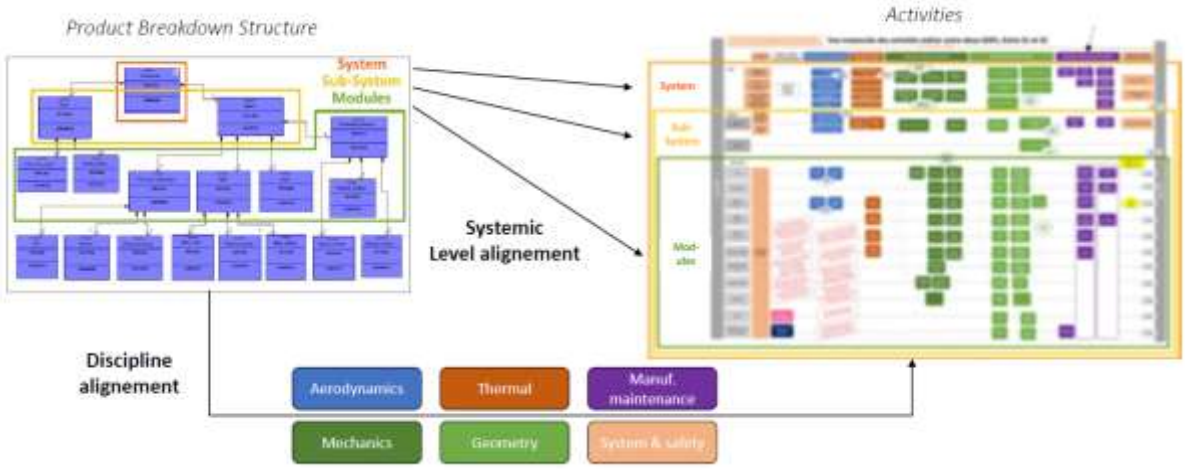

Figure 2 – Agile organization resulting from a product / organization alignment

2.2.2 Feature 2: Iterative Collaboration between Layers and Disciplines

The second idea of our agile approach for complex systems engineering consists in **iteratively defining the product** by organizing its development into **successive collaborative design loops** of increasing length and granularity where the product relevancy, consistency and feasibility is assessed at more and more precise levels of analysis (see next figure). Each loop shall in particular cover and study

several possible product variants at different levels of grain along their full development cycle, starting from top-level architecture up to manufactured part design and definition of manufacturing and maintenance concepts. The design choices shall be gradually frozen and deepened until achieving the full complete specification of the product and of its associated manufacturing and maintenance processes. All system stakeholders – in particular manufacturing & maintenance actors – shall be involved in these design loops in order to capture as early as possible their constraints.

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 & 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 say **all engineering & manufacturing disciplines** involved in a given product development – shall work collaboratively and share the same set of design drivers and hypotheses that will be progressively refined and frozen (see enabler 3). At the end of each design loop, a new product increment shall be generated through relevant models and tools (see enabler 4). Due to the progressive refinement of the design drivers, this approach provides a huge adaptability of the development process to new or updated requirements or constraints coming from various stakeholders. This approach also helps multidisciplinary teams to reach the global optimum choice of the product at each loop. The below Figure 3 illustrates such an approach with three design loops starting from a coarse grain high-level loop up to a fine grain detailed loop.

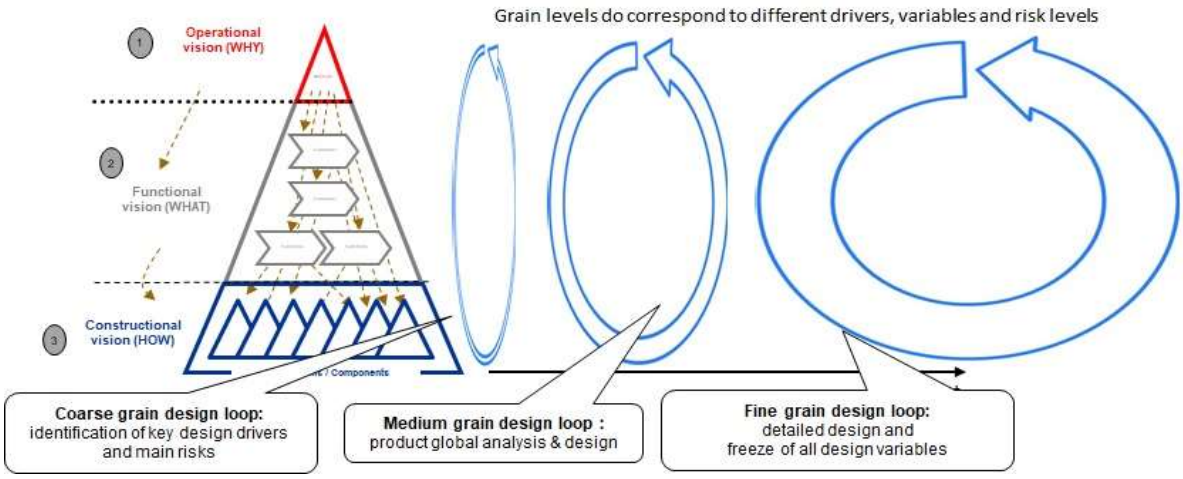

Figure 3 – Our agile development model based on successive and more & more detailed design loops

Each design loop will then analyze a number of design alternatives, starting typically with a dozen alternatives in the first loop, reducing them to 2-4 main alternatives in the second loop, up to converging to only 1-2 key alternatives in the last loop. This can be achieved by using trade-offs techniques, with different levels of details to reach a global optimum (see the below Figure 4).

Note that requirements are usually often over-specified and in much too important number, due to the fact that all actors in the development chain are creating requirements to protect themselves by taking local margins of security. In an agile approach, each design loop shall therefore pay a specific attention to formalize the system requirements to **what is just necessary**: this objective relies on prioritization techniques managed through collaborative requirements reviews, involving regularly all

key development actors, where allocations of performance are collaboratively managed and optimized when descending into product architecture.

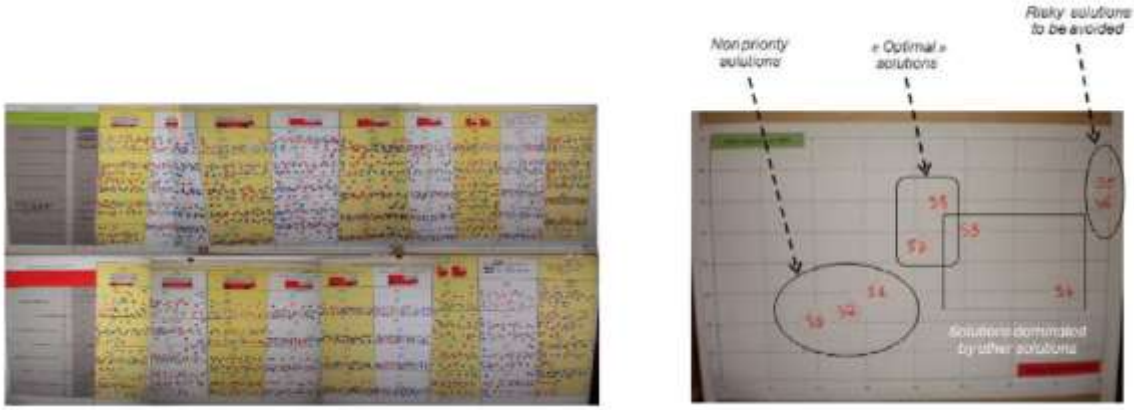

Figure 4 – Examples of several architectures compared through a Pareto multi-criteria optimization approach

Since we are dealing with systems, any agile development process shall also take into account 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 therefore be involved in each design loop – as already pointed out previously – in order to analyze design requirements from their perspective and to evaluate/simulate their impact on manufacturing and assembly processes with the "good" industrial KPIs (e.g. lead time, NRC & RC). Similarly support & services teams must be involved in order to evaluate the impact of design requirements on the maintenance processes using relevant KPIs (e.g. time and cost of maintenance operations).

Regular rituals, involving design, manufacturing and maintenance teams, and more generally all relevant stakeholders, shall therefore be regularly organized within each design loop in order to integrate all these different point of views in the design. Model-based systems engineering plays a key role in these rituals since they 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 (see an automotive example in the next figure).

Figure 5 – Example of a development ritual using model-based representations of a complex system

2.2.3 Feature 3: Iterative Key Design Driver's Management

The third key idea is the **preliminary identification of the key design parameters** on which each design loop rely. They shall form for each design loop a coherent set of interdependent characteristics, dealing with either product environment (e.g. mission profiles, use cases, customer & stakeholder requirements), either functional (e.g. presence of specific functions, behavioral performance) or constructional (e.g. mass, geometry of critical parts, technology) requirements, or manufacturing & maintenance constraints. These key design drivers also allow to explore design space and to identify potential architecting variants. The first loop focuses for instance on the structuring parameters of the product to analyze with highest priority. Note finally that these drivers can be derived from the analysis of a stakeholder need and functional & constructional requirement architecture, which are thus the first key models on which relies our agile approach.

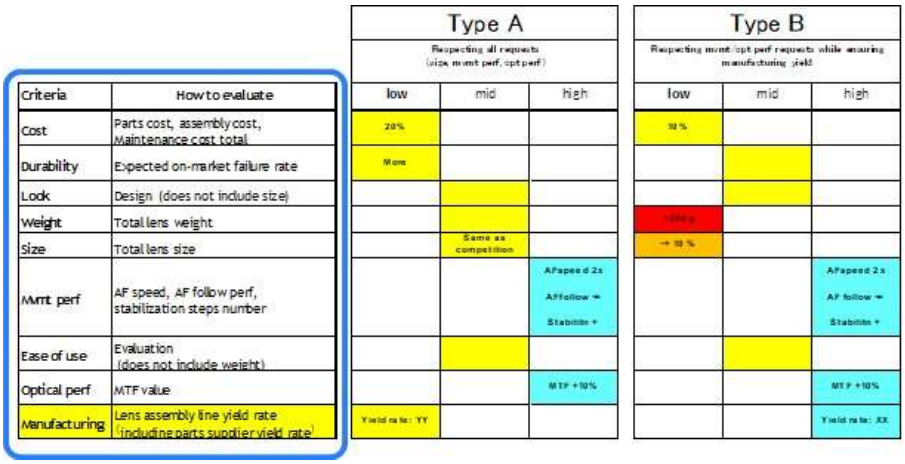

Figure 6 – Examples of design drivers identified in the context of an innovative camera development, here used to define two product variants that were analysed during a first design loop

The Key Design Drivers (KDD) represent the hypotheses shared between all teams involved in an agile development, i.e. the shared design space. During a given design loop, teams work with a fixed range for each KDD, with freedom of design while their parameters are contained into the defined ranges, from each synchronization ritual to the next one. At each iteration, teams define and evaluate potential architecture variants within KDD ranges in order to progressive reduce these ranges at the end of the iteration. If no feasible architecture can be defined within the design space defined by the current KDDs, a synchronization activity will redefine new KDD ranges.

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

2.2.4 Feature 4: DMU and FMU as Pivots for Transversal Synchronization

The last ingredient of an agile systems engineering process is to **use Digital & Functional Mockup (DMU and FMU) as pivots for transversal synchronization** in each design loop. The key idea is that at the end of each iteration, each team shall produce models (3D, thermal, mechanical resistance, thermodynamics, etc.) which are to be integrated in a shared DMU (for the geometrical aspects) and in a shared FMU (for the functional aspects), whose overall coherence is analyzed. Due to the need of

synchronizing all disciplines at each product increment, the use of these digital models as pivots for the design is therefore key to support an agile approach.

Note that usually the functional digital mockup is not a single tool, but rather a series of interconnected tools (e.g. Capella, Simulink, FL Editor, etc.). A key difficulty is here to ensure the alignment of the system decomposition used in the functional digital mockup with the geometrical decomposition on that the digital mockup manages, which currently is still a matter of human expertise. Both tools have indeed different modeling constraints and traditionally have been imposed separately. Nevertheless, to implement agile engineering, functional and discipline integration through both functional oriented design tools and CAD is a key enabler.

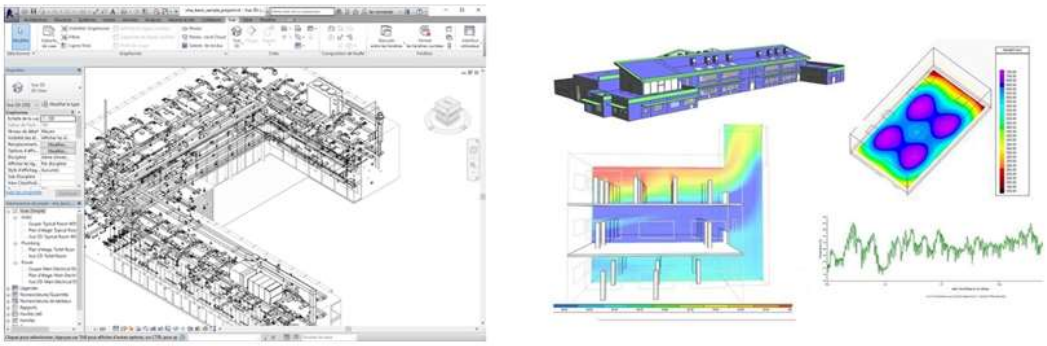

Figure 7 – Illustrations of Digital and Functional Mockup views

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

3. Conclusion

The CESAF framework has 4 key features, allowing an industrial company to successfully implement an agile / iterative systems engineering framework. These features are:

1. Product and organization alignment,
2. Iterative collaboration between layers and disciplines,
3. Iterative key design driver's management,
4. DMU and FMU 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.

References

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

[2] Aiguier M., Golden B., Krob D., *Complex Systems Architecture and Modeling*, [dans Posters of the first international conference on "Complex Systems Design & Management" (CSDM 2010), M. Aiguier, F. Bretaudeau, D. Krob, Eds.], 16 pages, 2010

[3] Aiguier M., Golden B., Krob D., *Modeling of Complex Systems II : A minimalist and unified semantics for heterogeneous integrated systems*, Applied Mathematics and Computation, **218**, (16), 8039-8055, doi : 10.1016/j.amc.2012.01.048, 2012

[4] Aiguier M., Golden B., Krob D., *An adequate logic for heterogeneous systems*, [dans Proceedings of the 18th International Conference on Engineering of Complex Computer Systems (ICECCS' 2013), Y. Liu, A. Martin, Eds.], IEEE, 2013

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

[6] ANSI/IEEE, *ANSI/IEEE 1471-2000 – Recommended Practice for Architecture Description of Software-Intensive Systems*, 2000

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

[8] Aslaksen E., Belcher R., *Systems engineering*, Prentice Hall, 1992

[9] Berrebi J., Krob D., *How to use systems architecture to specify the operational perimeter of an innovative product line ?*, [dans "Proceedings of INCOSE International Symposium of 2012 (IS 2012)", M. Celentano, Ed.], 15 pages, INCOSE, 2012

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

[11] Bliudze S., Krob D., *Towards a Functional Formalism for Modelling Complex Industrial Systems*, [dans ``European Conference on Complex Systems'' (ECCS'05), P. Bourgine, F. Képès, M. Schoenauer, Eds.], (article 193), 20 pages, 2005

[12] Bliudze S., Krob D., *Towards a Functional Formalism for Modelling Complex Industrial Systems*, ComPlexUs, Special Issue : Complex Systems - European Conference - November 2005 - Selected Papers - Part 1, **2**, (3-4), 163-176, 2006

[13] Bliudze S., Krob D., *Modeling of Complex Systems - Systems as data-flow machines*, Fundamenta Informaticae, Special Issue : Machines, Computations and Universality, **91**, 1-24, 2009

[14] Booch G., Jacobson I., Rumbaugh J., *The Unified Modeling Language Reference Manual,* Second Edition, Addison-Wesley, 2004

[15] Börger E., Stärk R., *Abstract state machines*, Springer, 2003

[16] Caseau Y., Krob D., Peyronnet S., *Complexité des systèmes d'information : une famille de mesures de la complexité scalaire d'un schéma d'architecture*, Génie Logiciel, **82**, 23-30, 2007

[17] CESAMES, *CESAM: CESAMES Systems Architecting Method – A Pocket Guide*, 134 pages, CESAMES, January 2017

[18] Cha P.D., Rosenberg J., Dym C.L., *Fundamentals of modeling and analyzing engineering systems*, Cambridge University Press, 2000

[19] Chalé Gongora H. G., Doufene A., Krob D., *Complex Systems Architecture Framework. Extension to Multi-Objective Optimization*, 3rd international conference on "Complex Systems Design & Management" (CSDM 2012), Y. Caseau, D. Krob, A. Rauzy, Eds.], Springer Verlag, 105-123, 2012

[20] Dauron A., Douffène A. , Krob D., *Complex systems operational analysis - A case study for electric vehicles*, [dans Posters of the 2nd international conference on "Complex Systems Design & Management" (CSDM 2011), O. Hammami, D. Krob, J.L. Voirin, Eds.], 18 pages, 2011

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

[22] de Weck O., Krob D., Liberti L., Marinelli F., *A general framework for combined module- and scale-based product platform design*, [dans Proceedings of the "Second International Engineering Systems Symposium", D. Roos, Ed.], 15 pages, MIT, 2009

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

[24] 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

[25] Doufène A., Krob D., *Sharing the Total Cost of Ownership of Electric Vehicles : A Study on the Application of Game Theory*, [dans Proceedings of INCOSE International Symposium of 2013 (IS2013)], 2013

[26] Doufène A., Krob D., *Model-Based operational analysis for complex systems - A case study for electric vehicles*, [dans Proceedings of INCOSE International Symposium of 2014 (IS2014)], 2014

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

[28] Douglas B.P., *Agile Systems Engineering*, Morgan Kaufman, 2015

[29] Friedenthal S., Moore A.C., Steiner R., A Practical Guide to SysML : the Systems Modeling Language, Morgan Kaufmann OMG Press, 2012

[30] Giakoumakis V., Krob D., Liberti L., Roda F., *Optimal technological architecture evolutions of Information Systems*, [dans Proceedings of the first international conference on "Complex Systems Design & Management" (CSDM 2010), M. Aiguier, F. Bretaudeau, D. Krob, Eds.], 137-148, Springer Verlag, 2010

[31] Giakoumakis V., Krob D., Liberti L., Roda F., *Technological architecture evolutions of Information Systems : trade-off and optimization*, Concurrent Engineering : Research and Applications, Volume 20, Issue 2, 127-147, 2012

[32] Grady J.O., *System Requirements Analysis*, Elsevier, 2006

[33] Grady J.O., *System Verification – Proving the Design Solution Satisfies the Requirements*, Elsevier, 2007

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

[35] Honour E.C., *Understanding the value of systems engineering*, INCOSE 2014 International Symposium, Vol. 14, 1207–1222, Toulouse, June 20-24, 2014, France, INCOSE, 2004; accessible at http://www.seintelligence.fr/content/images/2015/12/ValueSE-INCOSE04.pdf

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

[37] INCOSE, *Systems Engineering Handbook, A guide for system life cycle processes and activities*, INCOSE, January 2011,

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

[39] ISO/IEC/IEEE, *ISO/IEC/IEEE 15288:2015 – Systems and software engineering -- System life cycle processes*, May 2015

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

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

[42] Krob D., *Modelling of Complex Software Systems : a Reasoned Overview*, [dans ``26th IFIP WG 6.1 International Conference on Formal Methods for Networked and Distributed Systems" (FORTE'2006), E. Najm, J.-F. Pradat-Peyre, V. Viguié Donzeau-Gouge, Eds.], Lecture Notes in Computer Science, **4229**, 1-22, Springer Verlag, 2006 (Invited speaker)

[43] Krob D., *Architecture of complex systems : why, what and how ?*, [dans Proceedings of ``COgnitive systems with Interactive Sensors (COGIS'07)", H. Aghajan, J.P. Lecadre, R. Reynaud, Eds.], Stanford University, 1 page, 2007 (Invited speaker)

[44] Krob D., *Comment sécuriser la conception et le déploiement des logiciels métiers par une démarche d'architecture collaborative ?*, [in ``Journée Française des Tests Logiciels", B. Homès, Ed.], 23 pages, CFTL, 2008

[45] 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

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

[47] Krob D., *Eléments de modélisation systémique*, [in L'Energie à découvert, R. Mossery - C. Jeandel, Eds.], CNRS Editions, 2013

[48] Lamport L., *Specifying systems – The TLA+ language and tools for hardware and software engineers*, Addison-Wesley, 2003

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

[50] Marwedel P., *Embedded system design*, Kluwer, 2003

[51] Meinadier J.P., *Ingénierie et intégration de systèmes*, Lavoisier, 1998

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

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

[54] *NASA, Systems Engineering Handbook, 2007-edition, NASA/SP-2007-6105, 2007*

[55] Parnell G.S., Driscoll P.J., Henderson D.L., *Decision marking in systems engineering and management*, Wiley, 20011

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

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

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

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

[60] Severance F.L., *System modeling and simulation – An introduction*, Wiley, 2001

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

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

[63] Simpson T.W., Siddique Z., Jiao J.R., *Product platform and product family design, Methods and applications*, Springer Verlag, 2006

[64] The Open Group, *TOGAF® Version 9.1 – The Book*, The Open Group, 2011

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

[66] Valerdi R., *The Constructive Systems Engineering Cost Model (COSYSMO), Quantifying the Costs of Systems Engineering Effort in Complex Systems*, VDM Verlag Dr. Muller, 2008

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

[68] Wilipedia, *Agile software development,* https://en.wikipedia.org/wiki/Agile_software_development

[69] Wikipedia, *COSYSMO*, https://en.wikipedia.org/wiki/COSYSMO

[70] Wikipedia, *Scaled Agile Framework,* https://en.wikipedia.org/wiki/Scaled_agile_framework

[71] Wikipedia, *Systems architecture*, https://en.wikipedia.org/wiki/Systems_architecture

[72] Wikipedia, *Systems engineering*, https://en.wikipedia.org/wiki/Systems_engineering

[73] Wikipedia, *Systems theory*, https://en.wikipedia.org/wiki/Systems_theory

[74] Wikipedia, *Trade-off*, https://en.wikipedia.org/wiki/Trade-off

[75] Wikiquote, *Systems engineering*, https://en.wikiquote.org/wiki/Systems_engineering

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

[77] Zeigler B.P., Praehofer H., Kim T.G., *Theory of modeling and simulation – Integrating discrete events and continuous dynamic systems*, Academic Press, 2000