The Role of Formal Methods and Continuous Monitoring in the Verification of Complex Air Transportation Systems

Wilson N. Felder[1]
Stevens Institute of Technology, Hoboken, NJ 07030

For a certain class of aerospace systems, such as the U.S. National Airspace System, it is theoretically impossible to fully verify the performance of the system in accordance with its requirements. This can be demonstrated as a consequence of basic system theory, and established in individual cases through the application of formal methods of verification. These systems are termed "complex," and require an additional element of life cycle performance monitoring to protect against unpredictable but inevitable emergent system behaviors. A protocol for the application of continuous monitoring is proposed as an amplification of a previously described framework for the design, development, test and operation of complex aerospace systems.

I. Introduction

This paper reports on the clarification and extension of the use of formal methods and continuous monitoring as part of a general framework for design, development, test, and operation of complex aerospace systems, as described earlier (Felder, 2016b). For some time, I have been advocating this approach, most recently in a special issue of *The ITEA Journal* (Felder, 2015). During the 2016 AIAA Aviation Forum I proposed a specific framework for the Verification and Validation (V&V) of air transportation systems, using the US National Airspace System (NAS) as a notional example. This paper extends and defines two elements of that framework, building on recent advances from the field of commercial software testing, and new findings from the application of formal methods to verification of digital avionics.

As a review, the framework proposed was composed of five key elements:

(1) Life cycle governance;
(2) Model based engineering;
(3) Iterative development;
(4) Continuous monitoring; and
(5) Embedded test tools and formal methods

The impetus for the development of this framework has been the ongoing conversation fostered by the AIAA Complex Aerospace System Exchange (CASE), which seeks to discover best practices for the design, development, test and operation of complex aerospace systems (of which the U.S. NAS is a notable example.)

Work on the elements of the framework continues, with contributions from numerous workers. Both model based engineering and the use of concurrent engineering as a tool for life cycle management were topics of discussion at CASE 2016, and are subjects for chapters in an upcoming book reviewing progress in complex aerospace system design, development, test and operation as reflected through the CASE meetings.

[1] Distinguished Service Professor, School of Systems and Enterprises, Member ATCA, AIAA Fellow

Indeed, the first three elements of the framework are increasingly seen as essential. As a review, it will be helpful to summarize the impetus for, and main aspects of, these three elements.

In this context, "life cycle governance" refers to the need for a continuous, integrated process that guides the evolution of a complex system from research right on through to disposal. This concept addresses our understanding that the speed of advancing technology, highly agile threats, and the continuous adaptive nature of requirements make the traditional stage gate process ineffective and insufficiently responsive (Szajnfarber & Weigel, 2012).

We understand "model based engineering" in a broad and historic context – not the narrowly constrained view that calls for use of a particular system level modeling tool such as SysML. Specifically, my preferred working definition is of a family of models of various types (real and fast time, human in the loop, virtual live and constructive) at various levels of fidelity, that serve as a channel of communication from later phases of earlier system iterations to earlier phases of later iterations.

Finally, the element of "iterative development" is to be understood as the application of the fundamental aspects of agile software development to the slower and lengthier cycles of the

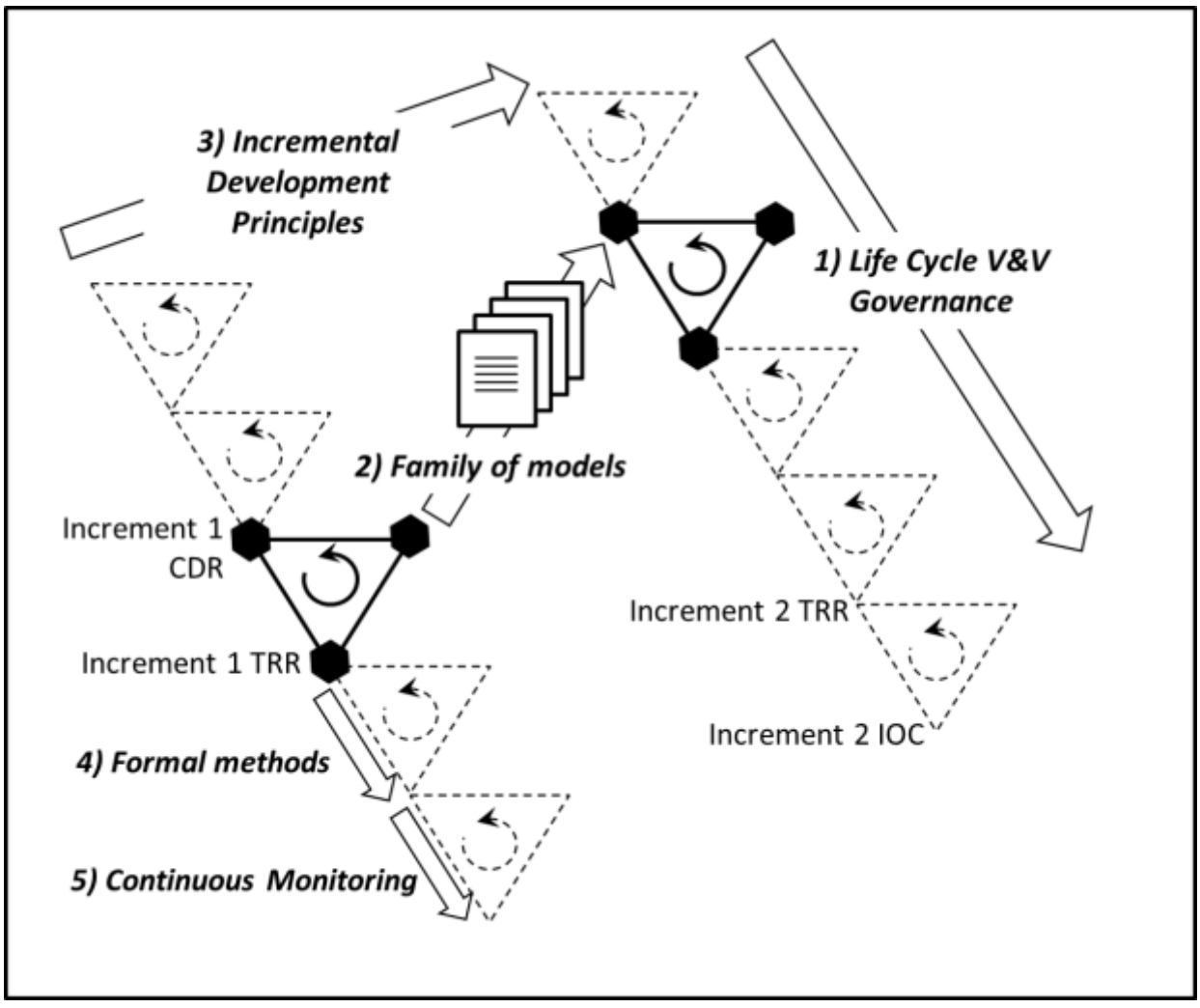

Figure 1: Proposed Framework for Design, Development, Test and Operation of Complex Aerospace Systems (after Felder (2016b) – as Amended)

traditional aerospace industry, which has been described as the incremental commitment spiral model (Boehm, Lane, Koolmanojwong, & Turner, 2014). This concept has been widely applied, both formally and by happenstance, but is recognized as a best practice in the high-speed world of complex system development.

The last two elements of the proposed framework have so far been less well defined, even though they were identified as key complex system verification methodologies at least as early as 2008 (Selberg & Austin, 2008), and in my previous description of the framework (Felder, 2016), were presented as more of a selection of "other" techniques rather than as distinct features. One objective of this paper is to further clarify and extend these ideas as part of the framework structure. Although both approaches are possible aspects of a verification and validation strategy, they differ significantly in their application and timing.

Figure 1 is a revision of the framework description from Felder (2016b) that clarifies the roles and positioning within the life cycle of formal methods and continuous monitoring as elements in the framework. In this view, embedded test tools become not an independent element, but just one aspect of a wide variety of techniques that together can be used to carry out the continuous monitoring function (as will be described in Section IV).

The purpose of this paper is to clarify the role of formal methods and continuous monitoring as part of the proposed framework of design, development, test and operations heuristics for complex aerospace systems. Along the way, I will review the theoretical foundation for the use of these two particular tools, incidentally suggesting a new definition of a complex system that is consistent with other definitions and makes clear the importance of formal methods and continuous monitoring.

II. Theoretical basis for the use of formal methods

We are still a long way from achieving the goal of mathematically defining complexity in the sense envisioned in John Casti's eloquent and beautiful (the only word for it) call to action in *Nature* (Casti, 2001). It is, however, possible to provide a practical answer to the question of what, exactly, a "complex system" is. Some authors have been more concerned with this question than others, and many have asserted that it doesn't matter (and in some sense, as we shall see, it doesn't), however, we shall also see that the resolution of the complexity question provides the rationale for the application of formal methods and continuous monitoring to the management of aerospace systems.

To see how this is so, consider a system of interest S defined in the usual manner, for example by:

$$S\colon P(\vec{I}) = \vec{O} \supset G\{o_m, o_{m+1}, \ldots, o_n\} \tag{1}$$

Which is just a restatement of von Bertalanffy's original 1950 definition of an open system (Von Bertalanffy, 1950), and which simply says that a system is defined by a process P that operates on a set of inputs $\vec{I}$ to produce a set of outputs $\vec{O}$ of which the subset $\vec{G}$ represents the goals of the system.

We now attempt to create a complete description of the system using some formal language such as the one proposed by Calisi *et al* and which is the subject of further refinement by Andrea Censi at MIT (Calisi, Censi, Iocchi, & Nardi, 2008). We will either succeed or not. If we cannot succeed for whatever reason – architectural or dynamic – then we know that we do not have a full and complete description of the system and in particular that we cannot *a priori* describe either its possible system states, or its behavior when presented with certain inputs (or time series of inputs).

Such a system is not amenable to complete verification prior to deployment, because we cannot know the full extent of its behaviors. We will use the term "complex" to describe such a system. On the other hand, suppose we succeed in building a description of the system using formal language. Such a description is now used to characterize the full response of the system to the complete spectrum of possible inputs (and time phased input strings…), usually by exhaustively tracing individual input values through the model formed by the formal language description of the system. If we are successful, then the design of the system will be "proved," that is, we will have complete understanding of the system's response to each and every possible input value (or time phased string of such values…). If we cannot complete this analysis (either because it's too hard, or for other technical reasons) then we will once again term the system "complex" because its future behavior cannot be predicted.

To put it another way, we will define a system as complex IFF:

1) It cannot be fully described using the syntax of a formal language; or
2) It can be so described but the resulting description cannot be solved.

This definition seems to ring true in the case of almost every example of complex systems that have been suggested in the past. It also admits of the existence of large, convoluted systems such as the Mars Science Laboratory Entry Descent and Landing system (MSL EDL) (Steltzner et al., 2010) and the National Ignition Facility (NIF) (Felder, 2016a), that probably *can* be described using formal language descriptions that are solvable in the formal methods sense. It also might explain why other systems such as the F-35 Joint Strike Fighter, which on the surface might appear simpler and more straightforward than either the MSL EDL or NIF, seem to present difficulties of emergence in their verification campaigns. (That is, they produce results that were not predicted by the design of the system.)

From this discussion, we can now see that any system capable of description using the syntax of a formal language, for which the resulting model can be solved, is amenable to being adequately verified using either formal methods or more traditional test methods during Development Test (DT) and Operational Test (OT). If it cannot be, on the other hand, then it is subject to unpredictable behavior during operation, and other methods must be employed to prevent damaging results from that (unpredictable but inevitable) behavior. In this way, formal methods and continuous monitoring can work together to provide a warning and a safety net for complex systems throughout the life cycle. Note that both these methods are fundamentally all about verification (that is, answering the question "did we build the system right?" as opposed to "did we build the right system?".)

Complex aerospace systems programs that do not follow this protocol: that is, they *neither* are proven using formal methods of verification at the time of Development and Operational Test (DT/OT); *nor* do they put in place a continuous monitoring regime, will inevitably encounter difficulties during operation. As has been previously noted (Felder & Collopy, 2012), the probability of such problems remains high throughout the life cycle because the nature of emergent behavior in complex systems follows power law distributions with fat tails, not the better-behaved Gaussian form, as explained for general audiences by Taleb in his excellent popular account (Taleb, 2007).

As a practical matter, of course, it hardly matters whether the formal methods test is actually applied (although as our facility with these techniques, and the availability of suitable tools, increases, it's probably a good idea to take this step): for the vast majority of digitally rich, distributed systems-of-systems with human interaction and high volume, stochastic, asynchronous digital communication among a changing membership of constituent systems, there will be no

easily solvable formal language description possible. As a result, continuous system monitoring is an essential requirement in almost every case. Always understanding, of course, that there are some quite large and convoluted systems (such as MSL EDL and NIF as previously mentioned) that buck this trend.

III. Formal methods in verification of complex aerospace systems

The fundamental concept in formal methods is the idea that a system can be fully described by a "clean algorithm" using some formal language. The formal description can then be proven through formal logic, the conclusion being that a system built according to the formal description is proven to work as designed in all possible cases. The use of formal methods in V&V has been widely reported (Easterbrook & Callahan, 1998). Cofer (2015) has provided an excellent review of the fundamental concepts of formal methods verification and its application to aviation systems. Kuhn *et al* (2002) provide a concise and comprehensive review of the various methods available for this purpose. Such algorithms are relatively straightforward to develop for simple systems. Applying formal methods to more complicated arrays of systems poses more difficulty. Researchers at Rockwell Collins have had significant success in developing formal proof methods for a wide variety of communications and avionics systems and are actively pursuing the application of formal proof methods to larger, distributed and networked examples. When we go beyond a simple, closely contained piece of digital avionics to a wider, networked environment, however, the problem becomes much harder.

Ultimately, the problem with the strict application of formal methods to real systems comes from the nature of the formal proof methodology itself. As described by Gödel's incompleteness theorem, any "sufficiently powerful" formal language is capable of asserting statements that cannot be proven to be true (or false) within the context of the language. In the present case, this means that we cannot be sure that a "clean algorithm" that is provable using formal proof methods can be drawn up for any given system. The importance of Gödel's theorem is not that it is a practical impediment to the application of formal methods of verification, but that because of it, we know that no sufficiently powerful formal language used to describe a given system is guaranteed to create a system description free of unprovable statements. Therefore, the second step, of proving the algorithm through analysis, is required. For a rigorous proof of Gödel's theorem, consult any standard textbook on metamathematics, for example, Kleene (1952). Hofstadter provides a much more readable and entertaining, although still sound, accounting in his *tour-de-force* (Hofstadter, 1980).

Researchers at the Software Engineering Institute (and elsewhere) have had success in applying stochastic approaches in conjunction with formal methods to allow the application of formal methods in less structured environments (Youssef, 2011). It is important to recognize, however, that in these cases, the "proof" aspect of formal methods is necessarily compromised. The stochastic approach essentially works by "pruning" the tree of potential system states, the resulting assertion being some variant of the statement "we cannot prove that the algorithm works in all cases, but we're pretty sure that it does, and we can in fact prove it in an important subset of the problem space." The use of stochastic extensions to formal methods is extremely powerful. It permits evaluations of specific aspects of system performance, such as cybersecurity, and gives good assurance that a particular system has a high probability of surviving a wide range of potentially damaging emergent events.

It should be noted that in Rockwell Collins' lengthy exploration of formal methods, "we have not yet run into Gödel as a limitation…." (Whiting, 2016) However, it is also true that Gödel

exists as a potential final barrier to the definitive, closed form determination of system verification. It is for this reason at least that an additional safety net, in the form of continuous system monitoring, is required. It is encouraging that formal methods, perhaps with the addition of stochastic pruning as described above, can at least provide a robust first pass of system verification.

An exciting opportunity for further study of this problem is found in Andrea Censi's work at MIT (Calisi et al., 2008). He has developed a system design tool that can be used to describe, in plain language, the system level requirements for a system, which are then converted into formal language. Software manages the process of description in such a way as to ensure completeness. The requirements are dynamically integrated into a top-level system design. The language is a formal one according to the definitions of formal logic, and is "sufficiently powerful" in the Gödel sense, so it provides an effective platform to determine theoretically whether any given system can, or cannot, be described in a way that makes it appropriate for the application of formal methods absent any additional pruning of system states. A potential experiment would involve coding a candidate system design using Censi's language. One of two things is then true: either (a) such a mapping is feasible, or (b) it is not. If (a), then it is by definition provable by the application of formal methods. If (b) it is not, and therefore will exhibit system states that have not been predicted and may result in damage to the system or unacceptable behavior on the system's part. (actually: if (a), then it is subject to formal proof, which may result in a situation where a given outcome may be provable or not, in which case, see (b)).

Even though in general formal methods are best suited for pre-deployment use, they can also be used to establish benchmarks for system observation during operations. An excellent example of the innovative use of this concept using the modeling tool AADL (developed by the Software Engineering Institute) is described by Hecht (2015).

In the end, however, there are at least three reasons that post-deployment monitoring is required, the first of which is the formal logic limitation described above. A second important factor is the reality that most systems of any interest to the aerospace community are embedded in complex socio-technical systems in which the presence of human operators forces complexity with the attendant presence of emergent behaviors, behaviors that regardless of their genesis, affect the operation of the engineered system and require response by that system. Finally, complex software rich systems exist in surroundings characterized by rapid environmental change, evolving challenges and threats, and emerging use cases. Figure 2 illustrates the application of formal methods and continuous monitoring to provide comprehensive verification over the life cycle.

Significant obstacles remain to the widespread implementation of formal methods, as pointed out by Hall *et al* (Hall, Driscoll, & Schweiker, 2012). Honour (Honour, 2013) too, has pointed out some of the difficulties in system verification that result from distributed systems in a system-of-systems, which is a commonly encountered scenario.

IV. Application of Continuous Monitoring

Formal methods are focused on the design of the system, and are intended to serve as proof that a particular system design will satisfy the system requirements in all possible cases. Continuous monitoring, however, which is also sometimes referred to as recurring surveillance, recognizes that in the case of certain systems of sufficient complexity, formal Development (DT) and Operational Test (OT) campaigns will be unable to fully explore the system's performance envelope, thereby allowing for the possibility of unpredicted (and perhaps unpredictable) failures during the system's operational life. This result is a certainty in the case where the evolving threat environment, spontaneous changes in system use, and other factors, result in unanticipated system

use (although strictly speaking, this can only be true if a given system fails to be provable through the application of formal methods, because if the formal methods proof is achievable, then no off nominal behavior is possible). It seems likely that most aerospace systems deployed prior to around the turn of the 21st Century would have been amenable to verification using formal methods, and that only since the widespread interchange of digital data among systems has there been an increase in the prevalence of systems that would be considered "complex" under the metric proposed in Section II. This is, of course, mere conjecture, as is the idea that the source of the complexity resides in the high volume, stochastic, asynchronous nature, and *ad hoc* (i.e. unscheduled) use of such digital interchange. It should be noted that it would be very likely possible to have a system made up of constituent parts (systems or subsystems) each of which could be proven using formal methods of verification, but whose interconnected whole could not be so characterized.

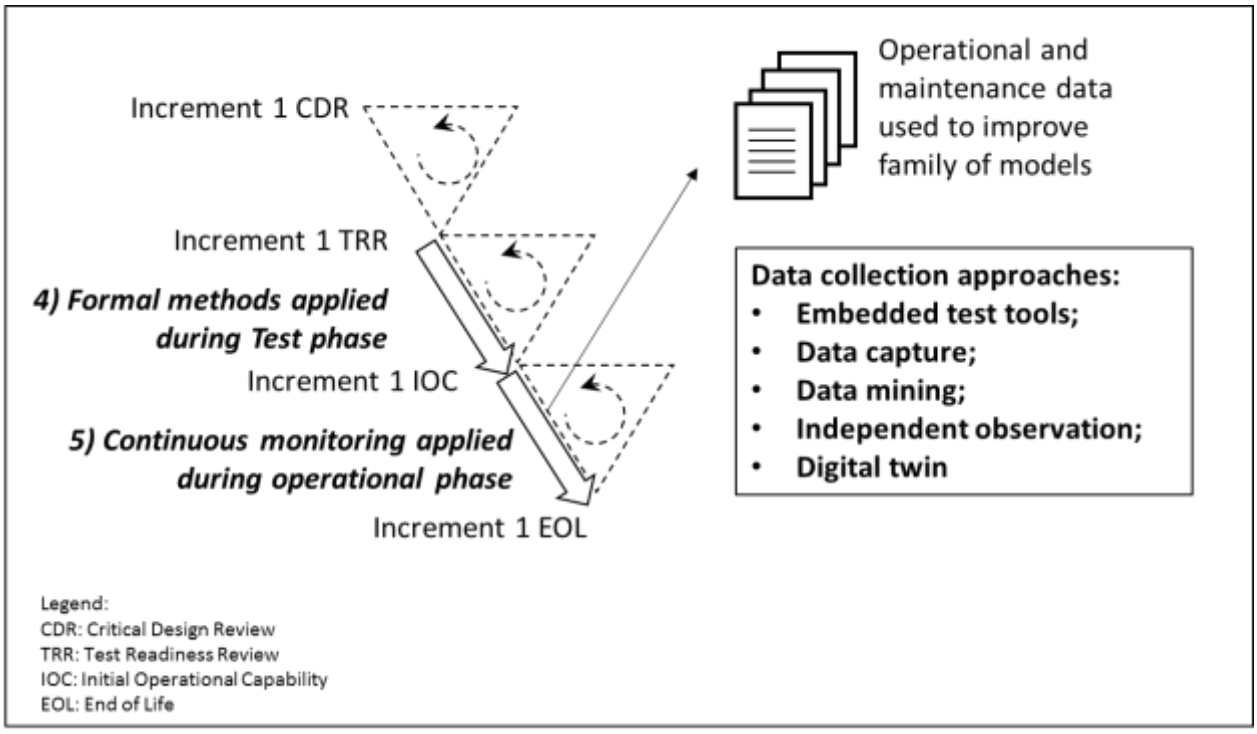

Figure 2: Detail of Framework showing use of Formal Methods and Continuous Monitoring to provide verification over the life cycle

In Figure 2 above a portion of the overall framework shown in Figure 1 is reproduced to clearly show the application of formal methods and continuous monitoring to the test and operation phases of the life cycle.

Formal methods are applied during Development Test (DT – between the Test Readiness Review and Initial Operating Capability milestones), with the objective of (1) providing as complete a "proof" based verification of the system as possible, and (2) establishing whether or not the system can be satisfactorily verified during the formal test phase, or whether continuous monitoring is required during the operational phase in order to protect from emergent effects.

Continuous monitoring is applied, on the other hand, during the operational phase (between Initial Operational Capability and End of Life) and has two major goals: first, to protect against unpredicted but inevitable emergent effects of a damaging nature, and second, to provide an ongoing stream of information to the family of models at the core of the framework in order to permit ongoing improvements to the fielded system (denoted by the circular arrow at the center of

the triangle) and to inform the design of the next increment in the system development. We have seen that in almost every case, we will be unable to fully verify the design of a system using formal methods, leading to the need for continuous monitoring of systems performance. In the context of the life cycle, it is best to think of continuous monitoring as a means to protect against that infinitesimally small exposure to unpredicted, and unwanted system states that exist, in a theoretical sense, as a result of the "pruning" process involved with the practical application of formal methods to system verification. Both maintenance and operational data can be used to improve the fidelity of the model(s) and the quality of the resulting designs.

Once we have concluded that continuous monitoring is necessary (and it will always be a useful component of the overall framework because it leads to improvements in the family of models used to characterize future system increments) the process itself can take several different forms, which can include:

- Embedded test tools;
- Monitoring of performance data;
- Analysis of data;
- Independent observation of system performance;
- Maintenance of a "digital twin".

A variety of continuous monitoring tools and techniques are currently used widely across industry and in the aerospace environment generally. All these are useful in the context of a complex system framework, with varying applicability depending on the nature of the system of interest. Figure 2 shows in detail how the various continuous monitoring techniques are employed in practice over the life cycle. A brief review of each of these approaches follows. These techniques have been used fairly widely in software maintenance, under the heading of "recurrent surveillance," as described generally in April *et a l*(2005) and are also reflected in the U.S. Air Force concept of the digital thread/digital twin initiative. I have previously described some of these in connection with the FAA's test programs (Felder, 2016b). These are summarized below for completeness.

At the most fundamental level, embedded test tools have been used widely in the commercial software environment. Under this approach, data is captured during software execution, and used to trigger various responses as required to correct emergent issues. The use of these approaches in aerospace has been less prevalent. Initial results under an ongoing NASA research initiative have proven promising. The use of embedded test is exemplified in the NASA cFS program as reported by McComas *et al* (2015).

A slightly higher level and much more widespread practice involves the collection of performance data on operational systems. This approach is very commonly used by airline maintenance organizations to monitor system performance aboard aircraft in near real time in order to schedule preventative and corrective maintenance. The data are downloaded to airline technical operations via the ACARS digital data link system.

The FAA has used continuous monitoring to ensure the satisfactory performance of several systems acquired as services. Because the verification of these systems through testing is proprietary, and therefore beyond the agency's visibility, performance data are routinely collected on the Wide Area Augmentation System (WAAS) (McHugh, 2005), the Automated Dependent Surveillance System – Broadcast (ADS-B) system, and the Flight Service Station program. The importance of continuous monitoring in these cases has also been the subject of Department of Transportation Inspector General reporting (Guzzetti et al., 2011).

The use of independent observation to monitor performance has been used with the FAA's Reduced Vertical Separation Minima (RVSM) program (Falk, Gonzalez, & Perez, 2010).

Although a vital part of the U.S. National Airspace System, this system has no ground infrastructure, and no Government owned software or hardware. In this instance, the agency uses a network of independent observation stations in which the altitude of the aircraft, as transmitted from the on-board precision altimeter, is compared to an independently measured altitude derived from primary radar triangulation, or other methods.

In addition to simply recording the data observed about a given system's performance, data mining and analysis can be used to derive secondary results relative to system operation. To a certain extent this is already being done by the airline technical operations systems described earlier. An intriguing application of this continuous monitoring concept has been described by Normann (2015).

In all of these cases, data collected from, measured on, or extracted from operational systems is available for use by the central collection of models that can no be used to develop fixes to ongoing maintenance issues, or to design the next increment in a family of systems. A distinctive use of continuous monitoring to improve the family of models used for design of future system increments is the Air Force Digital Thread concept as described by Kraft (2015) (Kraft, 2015), and by West and Pyster (2015). The Digital Twin concept is also extensively reviewed in the recently published review of findings from the series of CASE exchanges (Grieves & Vickers, 2016). The Air Force actually recognizes two linked concepts – the digital thread and the digital twin – where the "twin" is a powerful unitary model that mimics the physical system with high fidelity, and the "thread" is a framework provided to build, maintain and operate the "twin." The digital twin concept is prevalent in manufacturing where it has not generally been applied to complex systems. The "digital thread" idea is similar in intent to the FAA's family of models – sometime called a System of Systems Assessment Platform (SOSAP)– described in my earlier framework paper (Felder, 2016b).

V. Conclusions

The goal of this paper has been to clarify the roles of formal methods and continuous monitoring as elements in an overall framework for the design, development, test, and operation of complex aerospace systems. Each of these are intended as tools for verification of system performance but they act in different ways. Formal methods are a more theoretically based version of traditional development and operational test methods. If successful, they can be used to prove that a given system behaves as designed for all possible variations of system inputs. On the other hand, if no formal methods based verification process is possible, we are faced with a complex system for which we cannot predict all possible outcomes. In this case, a continuous monitoring regime is required to protect against unpredictable but inevitable emergent effects with potentially detrimental impact to the system.

With the greater definition of the role to be played by formal methods and continuous monitoring in the design, development, test and operation of complex aerospace systems such as the U.S. National Airspace System, we now have a much stronger and more robust proposed framework for the successful and confident deployment of operational systems. In addition, we have a means for determining whether a given system is likely to require continuous monitoring because of inherent complexity. The result, as depicted in Figures 1 and 2, represents "version 2.1" of the proposed framework.

References

April, A., Huffman Hayes, J., Abran, A., & Dumke, R. (2005). Software Maintenance Maturity Model (SMmm): the software maintenance process model. *Journal of Software Maintenance and Evolution: Research and Practice*, *17*(3), 197–223.

Boehm, B., Lane, J. A., Koolmanojwong, S., & Turner, R. (2014). *The incremental commitment spiral model: Principles and practices for successful systems and software*. Addison-Wesley Professional.

Calisi, D., Censi, A., Iocchi, L., & Nardi, D. (2008). OpenRDK: a modular framework for robotic software development. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 1872–1877). IEEE.

Casti, J. L. (2001). Formally speaking. *Nature*, *411*(6837), 527–527.

Cofer, D. (2015). Taming the Complexity Beast. *The ITEA Journal*, *36*(4), 313–318.

Easterbrook, S., & Callahan, J. (1998). Formal methods for verification and validation of partial specifications: A case study. *Journal of Systems and Software*, *40*(3), 199–210.

Falk, C., Gonzalez, J., & Perez, J. (2010). Using Automatic Dependent Surveillance-Broadcast Data for Monitoring Aircraft Altimetry System Error. In *Paper no. AIAA 2010-8165* (pp. 1–12). Toronto, Ontario Canada: AIAA.

Felder, W. N. (2015). System Engineering, Reliability, Life Cycle Support and Testing: a New Theory of Test for Complex Systems. *The ITEA Journal*, *36*(4), 259–263.

Felder, W. N. (2016a). Systems engineering best practices on the National Ignition Facility. Presented at the 2016 Conference on Systems Engineering Research, Huntsville, AL.

Felder, W. N. (2016b). The U.S. National Airspace System: a Model for Verification and Validation of Complex, Distributed Systems-of-systems. In *AIAA 2016-3152* (p. p.3152). American Institute of Aeronautics and Astronautics. Retrieved from http://dx.doi.org/10.2514/6.2016-3152

Felder, W. N., & Collopy, P. (2012). The elephant in the mist: What we don't know about the design, development, test and management of complex systems. *Journal of Aerospace Operations*, *1*(4), 317–327.

Grieves, M., & Vickers, J. (2016). Digital twin: Mitigating unpredictable, undesirable emergent behavior in complex systems. In *Transdisciplinary Perspectives on Complex Systems* (pp. 85–113). Springer.

Guzzetti, J. B., DeWeese, B., Ward, C., Bond, G., Thiel, H., Nossaman, A., & Azuolas, A. (2011). *FAA Oversight Is Key for Contractor-Owned Air Traffic Control Systems That Are Not Certified* (No. AV-2011-149,) (p. 21pp.). DOT OIG.

Hall, B., Driscoll, K., & Schweiker, K. (2012). Verification and validation of distributed flight critical systems. In *2012 IEEE/AIAA 31st Digital Avionics Systems Conference (DASC)* (p. 9D3–1). IEEE.

Hecht, M. (2015). Verification of Software Intensive System Reliabiity and Availability Through Testing and Modeling. *The ITEA Journal*, *36*(4), 304–312.

Hofstadter, D. C. (1980). *Godel, Escher and Bach: An Eternal Golden Braid*. New York: Random House.

Honour, E. (2013). Verification and Validation Issues in Systems of Systems. *arXiv Preprint arXiv:1311.3626*.

Kleene, S. C. (1952). *Introduction to metamathematics, D. Van Nostrand Company, New York*. New York: D. Van Nostrand Co. inc.

Kraft, E. M. (2015). HPCMP CREATE TM-AV and the Air Force Digital Thread. In *Proceedings of the 53rd AIAA Aerospace Sciences Meeting, AIAA 2015* (Vol. 42).

Kuhn, D. R., Chandramouli, R., & Butler, R. W. (2002). Cost effective use of formal methods in verification and validation. In *Foundations' 02 Workshop* (pp. 22–23).

McComas, D. C., Strege, S. L., Carpenter, P. B., & Hartman, R. (2015). Automated Test for NASA cFS. *The ITEA Journal*, *36*(4), 278–287.

McHugh, T. (2005, June). *WAAS Program Test Overview and In-Service Monitoring*. FAA WJHTC Media Briefing.

Normann, B. (2015). Continuous System Monitoring as a Test Tool for Complex Systems of Systems. *The ITEA Journal*, *36*(4), 298–303.

Selberg, S., & Austin, M. A. (2008). Toward an evolutionary system of systems architecture. In *Proceedings of Eighteenth Annual International Symposium of The International Council on Systems Engineering (INCOSE 2008), Utrecht, The Netherlands*. Citeseer.

Steltzner, A. D., Burkhart, P. D., Chen, A., Comeaux, K. A., Guernsey, C. S., Kipp, D. M., … others. (2010). Mars science laboratory entry, descent, and landing system overview.

Szajnfarber, Z., & Weigel, A. L. (2012). Managing complex technology innovation: the need to move beyond stages and gates. *International Journal of Space Technology Management and Innovation (IJSTMI)*, *2*(1), 30–48.

Taleb, N. N. (2007). The Black Swan: The Impact of the Highly Improbable.

Von Bertalanffy, L. (1950). The theory of open systems in physics and biology. *Science*, *111*(2872), 23–29.

West, T. D., & Pyster, A. (2015). Untangling the Digital Thread: The Challenge and Promise of Model--- Based Engineering in Defense Acquisition. *INSIGHT*, *18*(2), 45–55.

Whiting, M. (2016, October). Personal Communication. A forthcoming publication will be cited in the final version of this paper.

Youssef, H. (2011). *Verification and Validation of Complex Systems*. SAE Technical Paper.