Premature Allocation of Program Requirements to Suppliers

Bohdan W. Oppenheim[1,2], Loyola Marymount University

Abstract. The paper presents a discussion of the difficulties of formulating stable requirements early in complex engineering programs, and the severe consequences on program execution. The problems are caused by the need to seek political and funding support for the program. Formal classical Systems Engineering (SE) and Program Management (CSEPM) methodology is based on the assumption that the knowledge to anticipate all interfaces and create good requirements exists early in the program, and it is only a matter of working out the details to build extremely complex devices such as satellites, aircraft, refineries, nuclear power plants and high speed rail. The paper argues that this works well only for well-understood systems but it breaks down when the knowledge of what needs to be done still needs to be discovered, which is the case with most complex systems. In programs that develop new complex systems, the reality leads to the following Faustian Bargain: "Either develop and anticipate all interactions and requirements early, and allocate them to suppliers when the knowledge is not yet available, then conduct massive, painful, and cost-and schedule-busting requirements changes throughout the program; or delay the subcontracting until the system design is mature, complete and stable, and only then allocate requirements to subcontractors, but then risk the program termination because of the lack of political support and funding." The paper argues that in order to radically change this major deficiency of classical Systems Engineering and Program Management a radical change of the program business model would be needed.

1. Introduction

During the last 60 or so years, the classical Systems Engineering (SE) and Program Management (CSEPM) approach has evolved significantly, driven by two powerful forces: the uncompromising need for reliable system-level (including all subsystems) performance, and the inefficient government weapon acquisition practices. The evolution yielded a number of critical unintended consequences in CSEPM, including a shift from "great engineering" to "bureaucracy of artifacts", relying on massive outsourcing and inefficient mission assurance that involves premature requirement development, allocation and massive and costly requirements instabilities. The programs practicing CSEPM tend to achieve high levels of program success (e.g., 80 successful space launches in the U.S. Air Force [12]), but achieve this at the expense of notoriously costly and long (years and even decades) development programs, not infrequently with reduced performance.

This study is focused on requirements because they play a critical role in modern program formulation, execution and value/benefit delivery to customer stakeholders. It can be said that modern programs are driven by requirements. Yet, experience from complex programs such as satellite, spacecraft, ship, nuclear power plant, high speed rail, city infrastructure, and many others demonstrates that formulation of good and stable requirements is a formidable task and is rarely successful. In 2011 the Government Accountability Office [5] published an astonishing statistic that on average, 82% of requirements in recent defense programs are changed over the program lifecycle. That means that in spite of the huge effort, only 18% of the requirements released at the program initiation remain stable, a rather devastating number. This statistic is one reason for the notorious frustrations with large weapons and infrastructure programs, including exceeded program cost and schedule, Nunn-McCurdy reviews[3], and even premature program termination. Clearly, imperfect requirements are not the sole source of program troubles. Oehmen [9] lists 10 following Major Challenges in Managing Programs and each of them is capable of robbing a program of technical and/or business success:

1. Reactive Program Execution
2. Lack of stability, clarity and completeness of requirements
3. Insufficient alignment and coordination of the extended enterprise
4. Value stream not optimized throughout the entire enterprise
5. Unclear roles, responsibilities and accountability
6. Insufficient team skills, unproductive behavior and culture
7. Insufficient Program Planning
8. Improper metrics, metric systems and Key Program Indicators
9. Lack of proactive management of program uncertainties and risks
10. Poor program acquisition and contracting practices

Table 1 lists critical performance characteristics of nine major recent US Government space programs. The table data is based entirely on numerous GAO reports studied by [13]. The table indicates unstable requirements as a major contributor to program imperfect performance in seven of the nine programs listed. Besides unstable requirements GAO identifies the following other major reasons for program problems: unstable program funding (which is usually the result of other problems in a given program), starting the program before technology is sufficiently mature[4], and excessive complexity and features (named "gold plating of programs"[5] by Ashton Carter, then - Secretary of Defense for Acquisition and Logistics).

It is always desirable to correlate individual causes to program measures of success. Regretfully, the data quoted in Table 1 represents too small a sample size to allow that. The author was informed by his high-level contacts that that defense programs lack meaningful metrics of this kind. For example, government

"It is not realistic that all interfaces in a complex system can be anticipated and defined early in the program. Since all interfaces need be defined in order to write a complete set of requirements, it follows that it is not realistic to develop good detailed requirements at the program beginning."

PROG-RAM	Contr. Agency	Req's stable?	Funding stable?	# of TRL <<6	Final Cost B$	Cost Growth %	Schedule Growth%	# of Nunn-McCurdy Reviews	Excessive complexity?
SBIRS	Air Force	Unstable	Unstable	3	18.8	300%	120% Terminated	4	Yes
GPS IIF	Air Force	Unstable	Unstable	0	2.6	257%	133%	1+	No
GPS III	Air Force	Stable	Stable	0	4.2	2%	40%	0	No
GPS OCX	Air Force	Unstable	Unstable	14	3.695	28%	50%	N/A	Yes
MUOS	Navy	Stable	Stable	1	7.3	6%	20%	0	No
JMS	Air Force	Unstable	N/A	N/A	N/A	N/A	50%	N/A	Yes
SBSS	Air Force	Unstable	Stable	5	0.922	178%	60%	0	No
AEHF	Air Force	Unstable	Stable	11	14.372	154%	150%	3	No
NPOESS	Air Force, NOAA, NASA	Unstable	Stable	13	13.162	122%	Terminated	2	Yes

Table 1. Performance of Selected Major U.S. Space Programs [13]

programs do not track requirements instability over a program lifecycle, a critical measure of program quality[6]. The present paper is limited to a discussion of program requirements and related unintended consequences of the CSEPM evolution.

2.0 Unintended Emerging Properties of Classical Systems Engineering and Program Management

2.1 Premature Requirements and Massive Outsourcing

Developmental programs suffer from significant pressures to develop and allocate system requirements prematurely. The pressures are caused by the following two widespread practices:

a. Distribution of system development and production among as many geographically distributed suppliers as possible, driven by political pressures to "spread the wealth" in order to secure broad political support and funding for the program.

b. Overwhelmingly popular corporate policy to "stick to the system integration and subcontract the rest", with the vast majority of system parts built by a complex network of suppliers.

More specifically, prime contractors of modern weapons and aircraft perform system design, major structural design and systems integration, and subcontract subsystems and components to the established vendor base, e.g. Boeing 787, F-35. It is normal for supplier network that builds a complex system to include thousands of vendors in four tiers of suppliers. The heavily outsourced and geographically distributed programs make the program coordination and system integration challenging and increase the need for excellent CSEPM. In a structure such as this, contracts, requirements and specifications with suppliers perform a critical role. Requirements and specifications prescribe the technical performance and interfaces between multiple parties, and typically fixed-price contracts define the budgets and schedule for each supplier. This association of requirements with cost and schedule is a large activity of CSEPM. With such a distributed network of design and production, all linked by legal and financial contracts, the only way to effectively produce systems is to develop excellent top-level requirements, then flow down and allocate them into subsystem requirements, which then flow down and allocate these requirements into component requirements and "build-to" specifications. The critical issue is that politics and funding require that all this activity be performed early in the program, before detailed knowledge about the system has been developed. Thus, immature allocated requirements are contracted to the suppliers, and then the system developers frantically iterate the design and change the requirements - which is a source of major program instability. The critical assumption in this approach is that knowledge exists early in the program to anticipate all system interfaces and perform intensive development of requirements, requirement allocations to subsystems, and program planning, and then just execute the program in a single cycle of requirements-allocation-design-build-integrate-verify-and-validate in order to deliver extremely complex devices such as satellites, aircraft, refineries, nuclear power plants and submarines, and it is only a matter of providing enough resources to work out the iterations and details to create the needed system. The reality is that such formal techniques are effective only if designing a commodity for which significant legacy knowledge is available, but they break down when the knowledge of what needs to be done to write good requirements is lacking in early program phases, or the requirements are poorly formulated[7]. In this rigid system of thousands of fixed-price contracts with different suppliers in all tiers, any change of top-level requirements must be flowed down into all relevant suppliers and the contracts re-negotiated, typically with large delays, cost growth, or compromised performance in system development. The requirement verification and validation at all levels are the main tools of mission assurance in CSEPM. Unstable requirements and contested verifications are the notorious cause of arguments and legal actions between buyers and suppliers in the supply networks.

When a complex program starts with a large number of top-level requirements (and the recent trend is for increasing numbers, counted in low thousands), and when the technical knowledge of what is needed evolves over time slower than massive contracts with suppliers, the program instabilities cause significant “brute-force” iterations, information churning and thrashing due to the program pressures to keep requirements and test plans consistent – which tends to drive large cost and duration of complex programs. Almost all large governmental weapons programs demonstrate this behavior. The 82% of typical requirements being unstable mentioned earlier manifests that the single-pass execution of the classical SE process is not practical.

Hart-Smith [6][8] presented two additional powerful and well-substantiated arguments against massive outsourcing. Firstly, he documented the fact that outsourcing tends to outsource profits from a prime contractor to its subcontractors because subcontractors operate under fixed-price contracts; therefore the prime contractor has to absorb the costs of any design and requirement changes. Secondly, massive outsourcing introduces massive technical problems in system integration. This paper became quite controversial during the Boeing 787 aircraft development, which took the scope of outsourcing to extreme levels and experienced severe schedule and cost consequences.

2.2 Omitted Interfaces

In order to develop good requirements, all interfaces within the system and with system externalities must be identified. NASA SE Handbook [8] states (selected quotes from page 82):

“The bulk of integration problems arise from unknown or uncontrolled aspects of interfaces. Therefore, system and subsystem interfaces are specified as early as possible in the development effort. Interface specifications address logical, physical, electrical, mechanical, human, and environmental parameters, as appropriate....Interface specifications are verified against interface requirements...In verifying the interfaces, the system engineer must ensure that the interfaces of each element of the system or subsystem are controlled and known to the developers.”

With n elements in the system, there are n(n-1)/2 possible one-to-one interfaces. A typical space vehicle or craft has tens of thousands of elements. This alone makes the interface definition effort formidable, as each interface is needed to write good specifications. Practitioners of SE anticipating interfaces understand the trepidation question “have we included all of them?” - knowing that even one omitted interface may cause fatal failure.

Particularly challenging are the interfaces involving humans. Armstrong, [1] stated that “human beings are naturally wicked[9]; therefore interfaces with humans are inherently wicked.” In addition, most of the interfaces traditionally analyzed in technical systems are of the first order, with higher-order effects poorly understood and ignored. Two dramatic examples come from the two Space Shuttle tragedies. In the Challenger case, engineers understood that the rubber O-rings in the solid motor boosters must not be used in cold weather. They ignored the second-order human interface between the O-rings and the Shuttle flight management. The managers did not appreciate the risk of cold weather and ordered the flight, which led to the catastrophe, [2]. In the Columbia case, the interface between the foam covering the cryogenic tank and the airflow, as well as the secondary effect of the foam hitting and damaging the orbiter wings were poorly understood and ignored. The subsequent investigation determined that “the foam did it, the culture allowed it”, [3]. Both above interfaces involving “management” and “culture” qualify as “wicked human interfaces.” These interfaces were missing because the system was too complex to provide good insight and good understanding early in the program. And the more complex the system, the less knowledge is available just when it is needed.

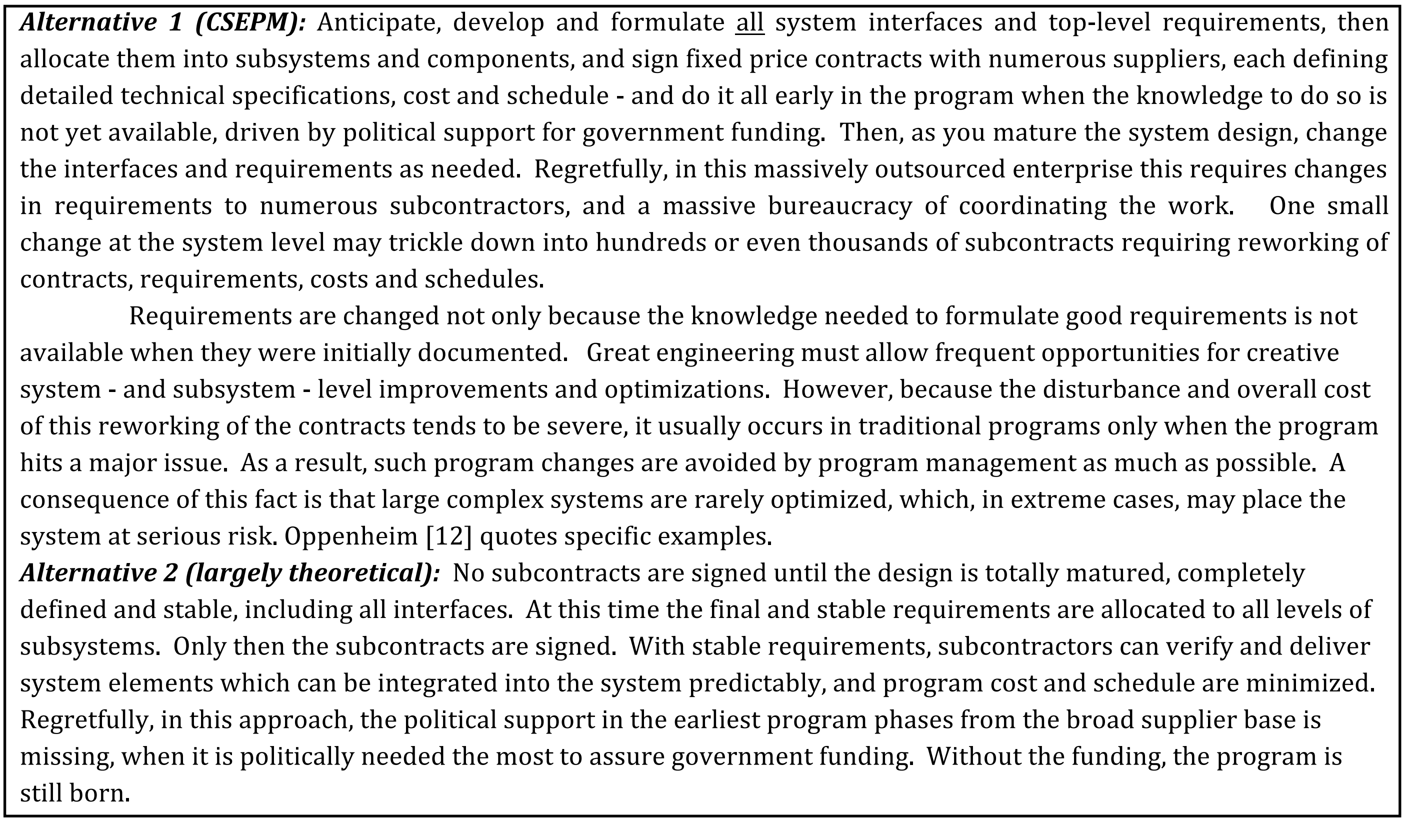
It is not realistic that all interfaces in a complex system can be anticipated and defined early in the program. Since all interfaces need be defined in order to write a complete set of requirements, it follows that it is not realistic to develop good detailed requirements at the program beginning.

2.3 Model Based Systems Engineering is not the Solution for the Interface and Requirements Instability

The recently introduced elegant Model Based Systems Engineering (MBSE) approach [7;4] strongly automate and facilitate interface and requirements management, dramatically reduce the time, cost, error rate and pain of the SE process, and offer a number of other significant benefits, but MBSE is a tool of CSEPM and suffers from the same fundamental problems as the CSEPM: it cannot assure that all interfaces have been properly included, particularly the “wicked” ones. MBSE can help in identifying possible interfaces by making the n-squared matrix easier to manage, but cannot fill in the insightful details in each matrix cell. That task is still left to the experience and intuition of engineers. The problem is that the experience and intuition work well only for well-understood systems. It is not realistic that all interfaces in a new complex system can be anticipated and defined early in the program.

2.4 Bureaucracy of Artifacts

Another unintended consequence of the massive outsourcing is the complexity of the effort needed to coordinate development and production among a large number of parties. CSEPM solves the problem by having different organizations create Interface



Alternative 1 (CSEPM): Anticipate, develop and formulate all system interfaces and top-level requirements, then allocate them into subsystems and components, and sign fixed price contracts with numerous suppliers, each defining detailed technical specifications, cost and schedule - and do it all early in the program when the knowledge to do so is not yet available, driven by political support for government funding. Then, as you mature the system design, change the interfaces and requirements as needed. Regretfully, in this massively outsourced enterprise this requires changes in requirements to numerous subcontractors, and a massive bureaucracy of coordinating the work. One small change at the system level may trickle down into hundreds or even thousands of subcontracts requiring reworking of contracts, requirements, costs and schedules.

Requirements are changed not only because the knowledge needed to formulate good requirements is not available when they were initially documented. Great engineering must allow frequent opportunities for creative system - and subsystem - level improvements and optimizations. However, because the disturbance and overall cost of this reworking of the contracts tends to be severe, it usually occurs in traditional programs only when the program hits a major issue. As a result, such program changes are avoided by program management as much as possible. A consequence of this fact is that large complex systems are rarely optimized, which, in extreme cases, may place the system at serious risk. Oppenheim [12] quotes specific examples.

Alternative 2 (largely theoretical): No subcontracts are signed until the design is totally matured, completely defined and stable, including all interfaces. At this time the final and stable requirements are allocated to all levels of subsystems. Only then the subcontracts are signed. With stable requirements, subcontractors can verify and deliver system elements which can be integrated into the system predictably, and program cost and schedule are minimized. Regretfully, in this approach, the political support in the earliest program phases from the broad supplier base is missing, when it is politically needed the most to assure government funding. Without the funding, the program is still born.

Figure 1 "Faustian Bargain" of CSEPM

Control Documents (ICD) to document the interfaces among the program and system elements and coordinate the development. The ICDs are often complex documents that require months of work by tens of individuals each. Often, because of this long time scales, an ICD turns out to be obsolete on arrival, because system changes have taken place while the ICD was being created. Some programs spend years and significant treasure on such information churning. Instead of spending program effort and time on technical system optimization, program employees create massive ICDs. For this reason, CSEPM is said to have deteriorated from the early emphasis on "great engineering" to unintended "bureaucracy of artifacts."

2.5 Faustian Bargain

To summarize, the critical problem present in practically all programs creating complex systems is that the knowledge about the system which is necessary to define the top level requirements and their allocations is not available until both the system design and program are quite mature, but this is in conflict with political support, funding priorities, and outsourcing trends of programs. This leads the CSEPM to a "Faustian Bargain" described in Figure 1, of two equally bad alternatives, which is the critical unintended emerging characteristic of the CSEPM evolution during the last 40 years.

3.0 The Remedy

Let us accept that in the present climate of large government contracts the political and funding pressures require the practice of "spreading the wealth" among massive number of geographically distributed suppliers, otherwise the program risks the lack of funding. This, however, should not mean that premature requirements and specifications must be allocated to the suppliers before the design is mature and stable enough, which is bound to cause massive requirement and program instability. Quite the opposite: the prime contractor should perform complete system development and design to the level of "built-to package", and only then allocate the requirements, interfaces and production specifications to the suppliers. The proposed remedy is to create at the program beginning preliminary but binding agreements with suppliers for future work, thus assuring political support and funding for a new program, but to hold off with passing detailed specification to them until the design, interfaces, and the allocated requirements and specifications are fully mature, optimized and stable. The agreements should include promissory notes defining a minimum level of effort to be defined at a later date. This way, the hugely destructive requirements instability will be avoided, and programs will be enabled to execute predictably, stably and at minimum cost and schedule. Further, completing the design with full freedom from the contracts with suppliers is conducive to system optimization.

The proposed remedy, among others, has been captured in the so-called Lean Enablers for Managing Engineering Programs (LEfMEP], [9]. The publication includes 326 best practices which promote value to the customer stakeholders and reduce waste. The practices have been developed by integrating Systems Engineering, Program Management and Lean. A detailed description of LEfMEP is beyond the scope of the present paper.

We should also mention a radical solution undertaken by the privately owned rocket and spacecraft maker SpaceX: to be totally vertically integrated and not reliant on suppliers, and totally co-located, [12]. As such, there is no need to allocate and verify any requirements to suppliers. This business model has demonstrated extraordinary gains in system quality, development time and cost, but it is not practical to apply it to large government programs which inherently involve the "spreading the wealth" policy.

The system design should be performed by co-located teams with towering competence in the domain. If outside expertise is needed for the design, it should be brought into the team, rather than subcontracted out. Hart-Smith presents ample evidence of the destructive impacts on program health and system integration if system design is subcontracted out in pieces. Under no circumstances should the early part of system design, when the need for coordination is the strongest, be subdivided and outsourced to numerous vendors. Doing so is equivalent to cutting one's brain into pieces, sending them out to remote vendors, and then expecting that the pieces will function as a working brain.

4.0 Summary and Conclusions

Formal classical SE and PM (CSEPM) methodology is based on the assumption that the knowledge to anticipate all interfaces and create good requirements exists at the program initiation, and it is a matter of working out the details to build extremely complex devices such as satellites, aircraft, refineries, nuclear power plants and high speed rail. This works well for well-understood systems but it breaks down when the knowledge of what needs to be done still needs to be discovered, which is the case with most complex systems. In real programs that develop complex systems, the reality is more reminiscent of the "Faustian Bargain: "Either develop and anticipate all requirements and interactions early, when the knowledge is not yet available, and then conduct massive, painful, and cost-and schedule-busting requirements changes throughout the program; or delay the subcontracting until the system design is mature, complete and stable, and only then allocate requirements to subcontractors, but then risk the program termination because of the lack of political support and funding." As described in the text, the average number of requirements changes in large programs is 82%, which indicates that the first path dominates in industry. This Faustian Bargain is the unintended emerging characteristic of the CSEPM evolution during the last 60 years, driven by the geographical distribution of programs among a vast number of suppliers, which, in turn, is motivated by the politics of "spreading the wealth" and assuring program funding.

The paper identified other myths of modern CSEPM, including the assumption that all system interfaces can be anticipated early; and the naïve belief that distribution of design and production over a massive network of suppliers can be effectively coordinated with massive Interface Control Document bureaucracy.

The unintended evolution of CSEPM continues to worsen as the complexity of modern systems increases at significant rates. The constant dynamic of need, innovation and change makes it increasingly improbable that detailed and stable requirements can be developed at a program's initiation. This observation applies not only to space and national security programs but to a vast array of other complex government and commercial technology and socio-technological programs, such as cyber security systems, finance, internet communication, energy, nuclear waste, education, global warming, transportation, medical systems, and many others. In many of these programs the rational approach is to delay subcontracting specifications until the system design is mature and optimized, and requirements are stable. In addition, the design should be handles by a co-located team for ease of coordination and optimization.

The high cost and schedule penalty of renegotiating subcontracts in order to accommodate changing requirements and system optimization are not the only arguments against massive subcontracting. As [6] discussed from the perspective of a prime contractor, two other reasons are that massive outsourcing of value creation to numerous suppliers tends to outsource profits from a prime contractor to its suppliers, and introduces massive problems in system integration by the prime. The Boeing 787 program is an excellent example of highly excessive subcontracting.

The remedy proposed in the paper is to create at the program beginning a preliminary but binding agreements with suppliers for future work, thus assuring political support and funding for a new program, but to hold off with the passing of detailed specification to them until the design, interfaces, and the allocated requirements and specifications are fully mature, optimized and stable. The remedy, among others, has been captured in the so-called Lean Enablers for Managing Engineering Programs (LEfMEP], [9]. This approach should vastly increase program efficiency and weapon affordability.

Acknowledgement

The original study on which this paper was based was performed under the Project Management Institute (PMI) contract 14108 and published in [12]. The author is grateful to Mr. Stephen Townsend of PMI for his permission to include long fragments from the study in the present paper. The author wishes to thank Mr. John Muratore of SpaceX for generous sharing of the information which was used in the PMI study and from which the present paper strongly benefitted.

ABOUT THE AUTHOR

Bohdan "Bo" W. Oppenheim is a Professor of Systems Engineering at Loyola Marymount University in Los Angeles, and founder and co-chair of INCOSE Lean Systems Engineering Working Group, the largest Group of INCOSE. He co-led the project developing Lean Enablers for Systems Engineering (LEfSE), and served as an expert in the joint INCOSE-PMI-MIT project developing Lean Enablers for Managing Engineering Programs, integrating Lean Systems Engineering with Lean Project Management. Dr. Oppenheim authored the book Lean for Systems Engineering with Lean Enablers for Systems Engineering (Wiley, 2011), co-authored The Guide to Lean Enablers for Systems Engineering (PMI, INCOSE, MIT-LAI, 2012), and co-authored Lean for Banks; Improving Quality, Productivity and Morale in Financial Offices. He worked for five years at Northrop, four years at Aerospace Corporation, and consulted numerous defense and aerospace companies in the U.S. and Europe. His engineering degrees include Ph.D. from Southampton, U.K.; Graduate Engineer's Degree from MIT; MS from Stevens Institute of Technology; and B.S. (equiv.) from Warsaw University of Technology in Aeronautics. His industrial experience spans space, offshore, mechanical engineering and software, including several major aerospace and commercial programs. His credits include five books, numerous journal publications, $2.5 million in externally funded grants, and a 40-year industrial experience. He served on the teams honored with two Shingo Awards (2012 and 2013), and INCOSE Best Product Award (2010), as well as INCOSE Service Award (2014) INCOSE Fellow, 2015. The author published two former articles in CrossTalk.

NOTES

1. Professor of Systems Engineering, Loyola Marymount University, Los Angeles, boppenheim@lmu.edu. First submitted December 7, 2014.
2. This paper is based on a larger study [12].
3. A Nunn-McCurdy review is a mandatory congressional review for viability of a program when it exceeds 25% growth from its original cost.
4. The effect of contracting a program before it is ready from the technology readiness point of view was published in [11]. Programs are not supposed to proceed to the so-called Milestone B (design) unless all technology items are at Level 6-7 or higher. Yet, for expediency reasons many programs are contracted with immature TRL. (Wikipedia page <http://en.wikipedia.org/wiki/Technology_readiness_level> defines the TRL scale).
5. "Gold plating" refers to the program or system features and options which are excessive and unnecessary.
6. At the beginning of this study, a high-level manager from one of the Federally Funded Research and Development Centers supporting the U.S. Air Force Space and Missile Command offered staff time and budget to extract statistics on stability of requirements over program life cycle in major government space programs. After months of trying, it turned out such metrics are not collected and cannot be obtained without major effort.
7. Examples of bad requirements include sloppy, unclear, or incorrectly formulated requirements; not contributing to program value/benefit; mutually conflicted; parochially-motivated, not applicable to the system (e.g., the well-known case of submarine-relevant requirements being inserted into a spacecraft program); "gold-plated"; and the requirements that mandate that a particular subsystem design be included in the system, thus making system optimization difficult. Ideally, a rigorous independent review of all requirements should be performed after they are collected and before the requirements are released for RFP or contract. Highly competent and independent reviewers should catch all instances of unneeded and faulty requirements, catch the missing interfaces, push back on "gold plated" requirements, and demand that all these deficiencies be corrected before the program proceeds to the RFP or contract. Unfortunately, this is hardly ever performed.
8. The paper is labeled "Boeing Proprietary", however, it was leaked and published on the Seattle Times webpage <http://seattletimes.nwsource.com/ABPub/2011/02/04/2014130646.pdf>, therefore now exists in the public domain.
9. In this context, "wicked" refers to "unpredictable, mischievous."

REFERENCES

1. Armstrong J. R., "System Integration: He Who Hesitates is Lost", INCOSE Int. Symp., Las Vegas, 2014
2. Challenger Commission, Report to the President by the Presidential Commission on the Space Shuttle Challenger Accident, June 6th, 1986, Washington, D.C.
3. Columbia Accident Investigation Board (CAIB) Final Report on Aug. 26, 2003.
4. Friedenthal S., Moore A., Steiner E., Practical Guide to SysML, Second Edition: The Systems Modeling Language, The MK/OMG Press, 2012
5. GAO, DEFENSE ACQUISITIONS, "Assessments of Selected Weapon Programs", GAO-11-233SP, 2011.
6. Hart-Smith L. J., "Out-Sourced Profits - The Cornerstone of Successful Subcontracting", Boeing Third Annual Technical Excellence (TATE) Symposium, St. Louis, MI, February 14-15, 2001
7. Long D., Scott Z., "A Primer for Model-Based Systems Engineering", VITECH, Apr 4, 2012
8. NASA Systems Engineering Handbook, NASA/SP-2007-6105 Rev 1
9. Oehmen, J. Editor, The Guide to Lean Enablers for Managing Engineering Programs, PMI-INCOSE-MIT LAI, 2012
10. Oppenheim, B. W., Lean for Systems Engineering with Lean Enablers for Systems Engineering, Wiley & Sons, 2011.
11. Oppenheim B. W., "Improving Affordability: Separating Research from Development and from Design in Complex Programs", Crosstalk Journal, 25th Anniversary Issue, V. 26, No. 4, July/August 2013.
12. Oppenheim B. W., "Program Requirements: Complexity, Myths, Radical Change, and Lean Enablers", Project Management Institute, Paper 14108, 2015.
13. Pabst R.G., "Analysis of Management Practices in U.S. Space Programs using GAO data, and Mapping to Lean Enablers", Systems Engineering Capstone Project, Loyola Marymount University, Los Angeles, 2014
14. Wikipedia, 2014a: <http://en.wikipedia.org/wiki/Technology_readiness_level>
15. Wikipedia, <http://en.wikipedia.org/wiki/Load_testing> last accessed June 21, 2014