

Air Force Institute of Technology

AFIT Scholar

Faculty Publications

Winter 2021

Agile Software Development: Creating a Cost of Delay Framework for Air Force Software Factories

J. Goljan

Jonathan D. Ritschel
Air Force Institute of Technology

Scott Drylie
Air Force Institute of Technology

Edward D. White
Air Force Institute of Technology

Follow this and additional works at: <https://scholar.afit.edu/facpub>



Part of the [Business Administration, Management, and Operations Commons](#), [Industrial Engineering Commons](#), and the [Software Engineering Commons](#)

Recommended Citation

Goljan, J., Ritschel, J. D., Drylie, S., & White, E. D. (2021). Agile Software Development: Creating a Cost of Delay Framework for Air Force Software Factories. *Air and Space Power Journal*, 35(4), 47–57. https://www.airuniversity.af.edu/Portals/10/ASPJ/journals/Volume-35_Issue-4/RM-Goljan.pdf

This Article is brought to you for free and open access by AFIT Scholar. It has been accepted for inclusion in Faculty Publications by an authorized administrator of AFIT Scholar. For more information, please contact richard.mansfield@afit.edu.

Cost-of-Delay

A Framework for Air Force Software Factories

JAMES GOLJAN

JONATHAN D. RITSCHEL

SCOTT DRYLIE

EDWARD WHITE



The Air Force software development environment is experiencing a paradigm shift. The 2019 Defense Innovation Board concluded that speed and cycle time must become the most important software metrics if the US military is to maintain its advantage over adversaries.¹ This article proposes utilizing a cost-of-delay (CoD) framework to prioritize projects toward optimizing readiness. Cost-of-delay is defined as the economic impact resulting from a delay in product delivery or, said another way, opportunity cost. In principle, CoD assesses the negative impacts resulting from changes to the priority of a project.

The cost-of-delay concept has been successfully employed in the private sector and has been suggested for use in military budget management.² But this concept requires tailoring to fit the unique nature of a public sector entity. To test a proof of concept for a new defense-centric cost-of-delay model, an Air Force research team engaged in a CoD process with Kessel Run.

Patterned after commercial-sector practices, Kessel Run eschews traditional software development techniques in favor of emergent, Agile principles.³ The 2019 Defense Innovation Board supported this change, stating, “DoD must move from waterfall and spiral development methods to more modern software development practices such as Agile, DevOps, and DevSecOps.”⁴ The goal of this transformation

is the global delivery of war-critical software through rapid feedback loops and a user-centered design.⁵ But changing to the Agile development environment has introduced new challenges that could undermine the Air Force's ultimate goal of maximizing a finite budget. Kessel Run currently relies on expert judgment calls rather than repeatable and verifiable quantitative methods to prioritize their various product lines.

The Case for Cost-of-Delay

Software, a ubiquitous component of our military systems, is vital to national defense. A 2010 National Research Council report stated the DOD software code increased by more than an order of magnitude every decade between the 1960s and the 2000s, equating to approximately 25 percent annual growth.⁶ A 2017 estimate projected an annual growth rate of 15–25 percent in the demand for developing and maintaining all defense software.⁷

The volume of code combined with the increasing complexity of integration is pressuring management to meet project objectives. Particularly, the schedule has caused concerns in recent years. The 2019 Defense Innovation Board concluded the capacity (or lack thereof) of the Department of Defense to rapidly develop and deploy effective software directly and negatively impacts the Department's ability to adapt and respond to threats.⁸

The delays to the F-35 delivery due to problems with software testing is just one notable example of how speed is key to mission readiness.⁹ The Department has thus begun to employ Agile software management as a technique that prioritizes timeliness. But by extension, a prioritization model that includes a component of timeliness may likewise need to be employed.¹⁰ This article proposes cost-of-delay.

The Department's adoption of CoD has been investigated before.¹¹ In the late 1990s, the Air Force explored CoD as part of the Air Force Cycle Time Reduction initiatives, but it did not gain traction.¹² At the time, the Department of Defense was largely employing a waterfall method of development. Traditional DOD development practices such as waterfall are done in sequential steps with long timelines, which permits time-consuming but possibly more robust prioritization techniques. For its simplicity, CoD lacks appeal in such an environment.

But now that the Department is employing Agile software development in some environments, it needs decision-making tools that can keep pace. Agile is characterized by reduced cycle times and continuous customer feedback. In this fast-paced environment, decision makers need a quick, defensible method with which to make trade-offs. Of note, discussions as to the merits of various software development approaches are outside the scope of this article.¹³ Rather, the De-

partment of Defense's shift (right or wrong) to Agile presents a new opportunity to evaluate how prioritization occurs.

Requirement prioritization methods in defense programs deserve discussion because the impact of these prioritization decisions reverberates throughout the defense portfolio and affects military readiness. One method commonly employed to organize the sequence and prioritization of work is first-in, first-out (FiFo).¹⁴ This method is frequently used in inventory management systems to ensure older products are used before new ones. But in a software development environment, FiFo may lead to inferior value or readiness.

Certain software requirements are more critical than others, and some requirements can quickly become obsolete due to the dynamic nature of software. As a result, recent software organizations have looked beyond FiFo to techniques that can speedily assess value. Two such applied approaches are the Kano and MoS-CoW models.

The Kano model uses teams to categorize software requirements into five classifications based upon the customer's needs.¹⁵ The MoSCoW method takes a similar approach but with different classification groupings.¹⁶ Both approaches are similar in that they qualitatively group requirements by the degree of customer need. Both models rely on the assessment of subject matter experts to create the groupings, but relying on these qualitative judgments is their greatest weakness.

Why is this an issue? Research in 2013 by Joshua J. Arnold and Özlem Yüce revealed a problem identified as the highest-paid person's effect (HiPPO).¹⁷ The HiPPO, typically the most senior individual in the room, remained adamant about the importance of certain requirements during the prioritization and planning stages. The study found eight other features appeared to be more valuable than the HiPPO's original choice. Clearly, overreliance on subject-matter-expert qualitative assessments can be problematic. Therefore, this article proposes CoD—a quantitative-based approach—as an alternative prioritization mechanism.

Anatomy of Cost-of-Delay

The CoD concept originated from Donald G. Reinertsen's seminal work quantifying the value of development speed.¹⁸ Reinertsen found a six-month delay can be worth 33 percent of life-cycle profits.¹⁹ These fundamental insights—time is valuable and quantitative economic analysis can improve decision-maker intuition—sparked a commercial-sector emphasis on lean product development and CoD implementation.²⁰ Over time, experimentation with CoD analyses in comparison to other methods revealed important insights. More specifically, the comparisons revealed the value of time is *not* intuitive, and decision makers often ar-

rive at divergent conclusions in the absence of a formal CoD model.²¹ Thus the need for CoD modeling was established.

This article uses the 2013 research by Arnold and Yüce, which further developed this CoD construct, as the framework for the CoD model.²² The efficacy of their construct was recently demonstrated through application by the international container shipping company, Maersk SeaLand.²³ This construct consists of three components: *benefit type*, *urgency profile*, and *development duration*.

The first component includes four different benefit types—*increase revenue*, *protect revenue*, *reduce costs*, and *avoid costs*.²⁴ Benefits are categorized by features that increase sales, help retain the business of existing customers, improve efficiency, or prevent foreseeable future costs. Because this study focuses on software, our explanation of these four benefit types uses the software nomenclature “features” to describe a distinguishing characteristic of the software item. The Institute of Electrical and Electronics Engineers defines the term in IEEE 829.

Urgency profiles, the second component of the model, are used to understand the life cycle of benefits and effects of being late.²⁵ Urgency profiles are categorized as *short life-cycle peak affected by delay*, *long life-cycle peak affected by delay*, *long life-cycle peak unaffected by delay*, and *impact of external deadline*. Each urgency profile is depicted in figure 1.

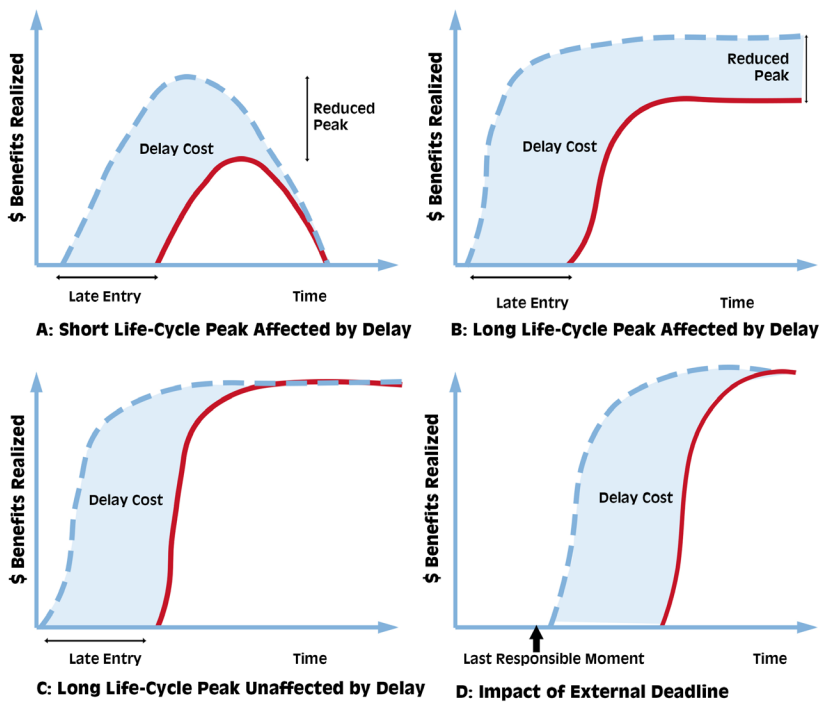


Figure 1. Urgency profiles

The third CoD component, *development duration*, is the amount of time necessary to complete a requirement. Combining a requirement's benefit type and urgency profile and dividing by the development duration produces a CoD score that can be compared to other requirements.²⁶

This calculation is a quantitative optimization framework to help prioritize requirements, tasks, or new work solely from a cost perspective. In economic terms, it is the opportunity cost between having some value now versus later. The opportunity cost is expressed as the dollar value that could be generated or saved per unit of time (days, weeks, months, etc.). To prioritize, the requirements with the highest opportunity costs per unit of time should be completed first.

A Public Sector Cost-of-Delay Model

The components of the CoD model outlined above must be modified for a public sector entity. An evaluation of the economic structure of government organizations and discussions with Kessel Run personnel concluded not all benefit types or urgency profiles were relevant. The benefit type *reduce cost* was included in the modified public-sector CoD model, and benefit types *increase revenue* and *protect revenue* were excluded. Urgency profiles *long life-cycle peak unaffected by delay* and *impact of external deadlines* were included in the modified public sector CoD model, and urgency profiles *long life-cycle peak affected by delay* and *short life-cycle peak affected by delay* were excluded.

The *increase revenue* and *protect revenue* benefit types are excluded from the model as a result of their association with profit generation. Due to the public goods nature of defense, Kessel Run and other public sector entities are not inherently revenue-seeking institutions, but *reduce cost* and *avoid cost* are relevant for a government setting.²⁷ *Reduce cost* covers changes that improve the overall efficiency of operations. *Avoid cost* consists of costs not currently incurred but may be in the future. Kessel Run's personnel identified both as the types of requirements or features their organization typically completes.

One urgency profile, *short life-cycle peak affected by delay*, is excluded from the public sector model. This urgency profile is identified when benefits are relatively short in duration and dictated quickly by market demand. For example, in the fashion industry, if a designer is late, the value of their commodity can be significantly reduced.²⁸ The assumption is DOD demand for certain capabilities typically will not fluctuate enough over short periods to warrant the consideration of this urgency profile.

The *long life-cycle peak affected by delay* profile is included in the public sector model. This urgency profile identifies features characterized by a clear first-mover advantage that penalizes latecomers.²⁹ It highlights benefits and costs associated

with falling behind rival competition—in the case of the Department of Defense, competition from other countries. The current US-China competition in space, in which the first mover would have the upper hand in a potential conflict, is characterized by this urgency profile.

The *long life-cycle peak unaffected by delay* profile applies to the Department as well, occurring when life-cycle benefits ramp up to a peak and are sustained over an extended period.³⁰ An example of this profile is process automation. The opportunity cost (measured in money per unit of time) is the same regardless of whether the acquisition is a first-mover or latecomer. All that matters is how many units of time it is available sooner, not which units of time. As a result, this urgency profile is the most common and easiest to compute.

The *impact of external deadline* urgency profile is also included in the model. In this configuration, a specific deadline is associated with a feature, and the CoD only begins to ramp up as it approaches the “last responsible moment.”³¹ To compute these profiles effectively, the team considers the lead time required to complete a particular feature and calculates an on-time delivery. Features that fall under this category are tied to a specific delivery date and will have a CoD of zero until the last responsible moment.

In summary, the CoD model can be modified for public sector use. Note, however, that the resulting CoD score should be considered in context with other available information. While the CoD will provide a quantitative, dollarized result for prioritizing requirements, other intangible benefits are not easily captured with a simple dollar estimate, for example, military or trade secrets. For this reason, the DOD cost-of-delay assessments are recommended as a complementary tool to help prioritize requirements but should not be considered a final, optimal solution in isolation.

Cost-of-Delay Model Test Case: Kessel Run

The test case for the CoD framework used data for analysis from two Kessel run application teams. The specific application teams—Chainsaw and Jigsaw—are part of Kessel Run’s operational command and control users product line. Each application team provided software features from their product backlog. For disclosure reasons, the exact specifications and descriptions of the features are not revealed. But both teams provided details regarding the work to be done as well as the potential cost savings to be gained from successful implementation.

The Chainsaw and Jigsaw teams used two features each for this analysis. In this simple model, it is assumed features are developed sequentially with no overlap. The four features analyzed identify reductions in manpower hours to determine


their cost-saving capabilities. The calculations considered included the *reduce cost* benefit type and followed the *long life-cycle peak unaffected by delay* urgency profile.

The opportunity cost was measured solely in terms of manpower costs, with Air Force Instruction 65-503, Table A33-1 providing the fiscal year 2020 hourly cost rates for active-duty military members used in the calculations. Table 1 provides the opportunity cost, development duration, and CoD scores for the four features provided by Chainsaw and Jigsaw. The research team prioritized the features with the highest CoD score resulting in the following order: Jigsaw Feature 2 (1140), Chainsaw Feature 1 (161), Jigsaw Feature 1 (152), and Chainsaw Feature 2 (24).

Table 1. Cost-of-delay scores for Kessel Run test case

Application Feature	Opportunity Cost	Development Duration (weeks)	CoD Score
Jigsaw Feature 1	\$456/week	3	152
Jigsaw Feature 2	\$1140/week	1	1140
Chainsaw Feature 1	\$483/week	3	161
Chainsaw Feature 2	\$24/week	1	24

The CoD scores in table 1 determine the order in which the four features should be undertaken. The CoD dollar value for the full set of features based on that prioritization required a second calculation. More specifically, the CoD incurred while developing Jigsaw Feature 2 was calculated as shown below (fig. 2).

$$= (\text{CoD JF 1} + \text{CoD JF 2} + \text{CoD JF 1} + \text{CoD JF 2}) * \text{Duration JF 2}$$


$$\$2103 = \left(\frac{\$456}{\text{week}} + \frac{\$1140}{\text{week}} + \frac{\$483}{\text{week}} + \frac{\$24}{\text{week}} \right) * 1 \text{ week}$$

Figure 2. CoD score = opportunity cost/duration

Note: Opportunity cost is a function of *benefit type* and *urgency profile*

Following this formula, the cost-of-delay incurred while working on Jigsaw Feature 2 was \$2,103. When working on Chainsaw Feature 1, since Jigsaw Feature 2 was already accomplished, the calculation only considered the CoD of Jigsaw Feature 1, and Chainsaw Feature 1 and 2. The Chainsaw Feature 1 CoD calculated as \$2,889 (\$456/week + \$483/week + \$24/week *3 weeks). Next, on the third prioritized feature, Jigsaw Feature 1, the CoD calculated as \$1,440 (\$456/week + \$24/week *3 weeks). Last, when working on Chainsaw Feature 2, the CoD was \$24. Adding these four CoD values together provided a total CoD of \$6,456, the lowest solution to this particular data set. Alternatively, had the team prioritized the features using a FiFo calculation, the total CoD would have been \$9,479.

The Kessel Run test case demonstrates several important points. First, the analysis reveals how some features are more significant than others from an opportunity-cost standpoint. Jigsaw Feature 2 and Chainsaw Feature 2 represent the greatest and smallest opportunity costs, respectively. Even with a small data sample, these results highlight the disparity that can be found when considering the importance of a product backlog.

Typically, a development team would focus on the features most important to the user. The assumption is the most important features will have the greatest operational opportunity costs. Therefore, CoD analysis provides a more quantitative and potentially more defensible way to illustrate which features are the most impactful to the user.

Second, these CoD assessments show how nonoptimal sequencing can add up to significant cost increases. For example, starting with the nonoptimal sequence of Chainsaw Feature 2—perhaps under the guiding principle “completing quick features”—would have yielded a large opportunity cost. Once again, the data set only represents a small sample of the potential cost saving. But with just this initial assessment, an increase in manpower efficiency from one feature can save the government thousands of dollars per week. A deeper discussion on the other cost-saving capabilities and the CoD quantification of the multitude of other features in the backlog could reveal even more efficiencies that could be achieved through the successful implementation of certain features.

Discussion

Cost-of-delay provides an organization with a methodology to optimize its portfolio's structure. To be clear, this article does not suggest CoD is a panacea. Rather, CoD is simply a quantitative method to improve decision making. It is important to note the opportunity for human mediation in the process is preserved. Agile development's flexible, iterative nature, coupled with intensive user feedback, ensures this mediation occurs.

While the CoD score establishes an initial means to prioritize features, leadership can adjust scores based on other subjective goals or those factors that directly impact war fighting and thus national defense readiness levels. Those gains must be considered in conjunction with the CoD model. What cost-of-delay adds to the current process is a quick, defensible framework through which decision makers can make better-informed trade-offs.

While this article provided the necessary framework for CoD implementation in the public sector, the demonstration of the CoD concept in a DOD organization is clearly limited. The duration of the prioritized features was short, and the dollar amounts were small. Yet this example should stimulate conversations in

organizations about the applicability of the CoD approach within the parameters of unique project or program characteristics.

The Air Force shift toward the Agile software development environment is the impetus to consider implementing novel CoD methodologies. The emphasis on valuing speed, cycle time, and user feedback lends itself to a CoD approach. The experience of the private sector provides sufficient evidence. The benefits to organizations are demonstrable in three areas: (1) making better decisions; (2) prioritizing in a way that maximizes value; and (3) changing the focus from efficiency and cost (which encourages wrong behavior) to speed and value.³² By tailoring the private-sector CoD model to the unique nature of a public sector defense organization, this study's Kessel Run test case suggests Air Force implementation is possible.

The positive outcomes experienced by the private sector directly translate to benefits for the war fighter. The war fighter gains from capability being delivered more quickly to the field, in part due to better decisions in the prioritization process. Cost-of-delay is one component that feeds the decision-making process. The magnitude of those benefits within larger projects will undoubtedly vary based upon specific circumstances. The suggestion from the data examined in this article indicates there is potential for large gains, but results must be caveated. Costs associated with gathering inputs to the CoD model, including the time required of the program manager and other subject matter experts to quantify impacts, were relatively low in this proof of concept. Yet those costs may rise and should be accounted for in a larger application of the CoD concept.

And as mentioned previously, CoD is a tool designed to provide value when prioritizing requirements. Implementing CoD will not alleviate all software development costs and schedule problems. Other models, such as cost of quality, that are constructed to help with some of these software development problems should be considered in conjunction with the CoD model.³³

The benefit from CoD is simple but important. It provides a cost-efficient approach to prioritizing features, once the program manager has determined the desired quality level of the software development. Thus, the utility of CoD to an organization should be evaluated within the context for which it was designed. Cost-of-delay provides one key piece of information to the decision maker but must be used in conjunction with other data when analyzing the holistic software development process.

The Kessel Run test case demonstrated in this article was important as a proof of concept. But it is only the beginning. Air Force software factories applying Agile techniques are emerging at a rapid rate. Larger-scale testing of the CoD concept in these USAF Agile development environments is warranted. Through

iterative feedback, organizations can modify and improve upon the CoD framework provided. If these endeavors are successful, future research should examine expanding the CoD concept for potential adoption by a wide range of other Air Force programs. ✪

James Goljan

Captain James Goljan, cost analyst at the Space Systems Center, Los Angeles AFB, CA, holds a master of science in cost analysis from the Air Force Institute of Technology, Department of Systems Engineering and Management.

Jonathan D. Ritschel

Dr. Jonathan D. Ritschel is an assistant professor of cost analysis in the Air Force Institute of Technology, Department of Systems Engineering and Management.

Scott Drylie

Lieutenant Colonel and Dr. Scott Drylie is an assistant professor of cost analysis at the Air Force Institute of Technology, Department of Systems Engineering and Technology.

Edward White

Dr. Edward White is a professor of statistics at the Air Force Institute of Technology, Department of Mathematics and Statistics.

Notes

1. J. Michael McQuade et al., *Software is Never Done: Refactoring the Acquisition Code for Competitive Advantage* (Washington, DC: Defense Innovation Board, March 21, 2019), viii, <https://media.defense.gov/>.
2. Donald G. Reinertsen et al., "An Overview of Cost-of-Delay Analysis: Calculating Project Decision Rules," *Journal of Cost Analysis & Management* 4, no. 1 (2002), <https://www.tandfonline.com/>.
3. McQuade et al., *Software is Never Done*, 10.
4. McQuade et al., *Software is Never Done*, 10.
5. "So You Want to Work for Kessel Run?," Kessel Run (website), accessed September 29, 2021, <https://kesselrun.af.mil/>.
6. National Research Council, *Critical Code: Software Producibility for Defense* (Washington, DC: National Academies Press, 2010), 19, <https://www.nap.edu/>.
7. David M. Tate, *Software Productivity Trends and Issues Conference Paper*, NS D-8365 (Alexandria, VA: Institute for Defense Analyses, March 2017), 1-2, <https://apps.dtic.mil/>.
8. McQuade et al., *Software is Never Done*, 1.
9. Government Accountability Office (GAO), *F-35 Joint Strike Fighter: Problems Completing Software Testing May Hinder Delivery of Expected Warfighting Capabilities* (Washington, DC: GAO, March 24, 2014), <https://www.gao.gov/>.

10. David Vergun, "Hicks Provides Overview of DOD Priorities," Department of Defense News, June 8, 2021, <https://www.defense.gov/>.
11. Reinertsen et al., "Cost-of-Delay."
12. Todd S. Butler, "Cost-of-Delay Analysis (CODA): Use and Implementation of This Decision Support Tool" (master's thesis, Air Force Institute of Technology, 2005), 16.
13. Barry Boehm, "Get Ready for Agile Methods, with Care," *Computer* 35, no. 1 (January 2002), <https://citeseerx.ist.psu.edu/>; S. Balaji and M. Sundararajan Murugaiyan, "Waterfall vs. V-Model vs. Agile: A Comparative Study on SDLC," *International Journal of Information Technology and Business Management* 2, no. 1 (June 2012), <https://moam.info/>; and Tsun Chow and Dac-Buu Cao, "A Survey Study of Critical Success Factors in Agile Software Projects," *Journal of Systems and Software* 81, no. 6 (June 2008), <https://www.sciencedirect.com/>.
14. H. M. Manohar and S. Appaiah, "Stabilization of FIFO System and Inventory Management," *International Research Journal of Engineering and Technology* 4, no. 6 (June 2017): 5631–34, <https://www.irjet.net/>.
15. "What is the Kano Model?, Kano Model (website), accessed September 30, 2021, <https://kanomodel.com/>; and Emmanuel O. C. Mkpojiogu and Nor Laily Hashim, "Understanding the Relationship between Kano Model's Customer Satisfaction Scores and Self-Stated Requirements Importance," *SpringerPlus* 5, no. 197 (2016), <https://doi.org/>.
16. Dai Clegg and Richard Barker, *Case Method Fast-Track: A RAD Approach* (Boston: Addison-Wesley Longman, 1994).
17. Joshua J. Arnold and Özlem Yüce, "Black Swan Farming Using Cost of Delay: Discover, Nurture, and Speed up Delivery of Value," in *2013 Agile Conference* (New York: Institute of Electrical and Electronic Engineers, 2013): 101–16, <https://doi.org/>.
18. Donald G. Reinertsen, "Whodunit? The Search for the New-Product Killers," *Electronic Business* 9 (July 1983): 62–65.
19. Reinertsen, "Whodunit?," 62.
20. Donald G. Reinertsen, *Managing the Design Factory: A Product Developer's Toolkit* (New York: Simon & Schuster, 1997).
21. Reinertsen et al., "Cost-of-Delay," 9–10.
22. Arnold and Yüce, "Black Swan Farming."
23. Arnold and Yüce, "Black Swan Farming."
24. Arnold and Yüce, "Black Swan Farming," 105–6.
25. Arnold and Yüce, "Black Swan Farming," 108.
26. Joshua J. Arnold, "Cost of Delay Divided by Duration," Black Swan Farming, January 5, 2014, <https://blackswanfarming.com/>.
27. Paul A. Samuelson, "The Pure Theory of Public Expenditure," *Review of Economics and Statistics* 36, no. 4 (November 1954): 387–89, <https://doi.org/>.
28. Arnold and Yüce, "Black Swan Farming," 108.
29. Arnold and Yüce, "Black Swan Farming," 108.
30. Arnold and Yüce, "Black Swan Farming," 108.
31. Arnold and Yüce, "Black Swan Farming," 108.
32. Ben Linders, "Using Cost of Delay to Quantify Value and Urgency," *InfoQ*, February 6, 2015, <https://www.infoq.com/>.
33. Joseph M. Juran, Leonard A. Seder, and Frank M. Gryna, *Quality Control Handbook*, 2nd ed. (New York: McGraw-Hill, 1962).

Disclaimer: The views and opinions expressed or implied in the Journal are those of the authors and should not be construed as carrying the official sanction of the Department of Defense, Air Force, Air Education and Training Command, Air University, or other agencies or departments of the US government. This article may be reproduced in whole or in part without permission. If it is reproduced, the Air and Space Power Journal requests a courtesy line.