NONSTATIONARY ROOT CAUSES OF COBB'S PARADOX

Lt Col Joseph W. Carl, USAF (Ret.)
and Col George Richard Freeman, USAFR (Ret.)

Cobb's Paradox states, "We know why [programs] fail; we know how to prevent their failure—so why do they still fail?" One possibility is that we do not really know why programs fail and there is no paradox. Another possibility is that some of the problems that lead to program failure may not be susceptible to practical solution, so that continued failure is not paradoxical. This article defines what we mean by nonstationary root causes of program failures, and identifies 10 such causes. Requirements volatility, funding stability, process immaturity, and lack of discipline are often cited among the reasons. The article ends with recommended approaches to mitigate the effects of influences from the environment that change over time—nonstationary effects.

Keywords: *Cobb's Paradox, Nonstationary Environments, Program Stability, Change Management, Configuration Management*

image designed by Miracle Riese »

In 2007, the many examples of government project failures led then-Under Secretary of Defense for Acquisition, Technology and Logistics John Young to issue a memorandum that requires prototyping and competition on all major programs up to Milestone B (Young, 2007). Young's memorandum was a propitious start. But is it likely to be sufficient to solve all the problems that lead to project failure?

This article summarizes the number and spectrum of project failures, and makes the case that project failures cannot be attributed solely to mismanagement on the part of project managers. Rather, it appears improbable that all project managers of large complex projects could produce similar failures. The prevailing perception throughout the acquisition community is that program and project managers know why projects fail and how to prevent them from failing. The authors discuss the concept of other influences from the environment that change over time—nonstationary effects—that may be the root cause of these numerous project failures.

Background

In 2006, a Government Accountability Office report (GAO, 2006) highlighted several government project failures.

> In the last 5 years, the Department of Defense (DoD) has doubled its planned investments in new weapon systems from about $700 billion in 2001 to nearly $1.4 trillion in 2006. While the weapons that DoD develops have no rival in superiority, weapon systems acquisition remains a long-standing, high-risk area. GAO's reviews over the past 30 years have found consistent problems with weapon acquisitions such as cost increases, schedule delays, and performance shortfalls.

The report goes on to state that this huge increase in spending over the past 5 years "has not been accompanied by more stability, better outcomes, or more buying power for the acquisition dollar." Examples of this huge increase in spending follow:

- Capable satellites, potential overrun of $1.4 billion
- Satellite payload cost and schedule overruns greater than $1.1 billion
- Radar contract projected to overrun target cost by up to 34 percent
- Advanced Precision Kill Weapon System (Joint Attack Munition Systems), curtailment of initial program in January 2005 due to development cost overruns, projected schedule

slip of 1–2 years, unsatisfactory contract performance, and environmental issues
- C-5 Avionics Modernization Program, $23 million cost overrun
- C-5 Reliability Enhancement and Re-engineering Program, $209 million overrun
- F-22A, increase in the costs of avionics since 1997 by more than $951 million or 24 percent, and other problems discovered late in the program.

On March 31, 2006, Comptroller General of the United States David M. Walker stated in congressional testimony:

> The cost of developing a weapon system continues to often exceed estimates by approximately 30 percent to 40 percent. This in turn results in fewer quantities, missed deadlines, and performance shortfalls. In short, the buying power of the weapon system investment dollar is reduced, the warfighter gets less than promised, and opportunities to make other investments are lost. This is not to say that the nation does not get superior weapons in the end, but that at twice the level of investment. DoD has an obligation to get better results. In the larger context, DoD needs to make changes…consistent with getting the desired outcomes from the acquisition process.

Cobb's Paradox

In 1995, Martin Cobb worked for the Secretariat of the Treasury Board of Canada. He attended The Standish Group's CHAOS University, where the year's 10 most complex information technology (IT) projects are analyzed and discussed. The 10 most complex IT projects studied by The Standish Group in 1994 were all in trouble: eight were over schedule, on average by a factor of 1.6 and over budget by a factor of 1.9; the other two were cancelled and never delivered anything. That led Cobb to state his now-famous paradox (Cobb, 1995): "We know why [programs] fail; we know how to prevent their failure—so why do they still fail?"

The Standish Group uses project success criteria from surveyed IT managers to create a success-potential chart. The success criteria are shown in the Table, where they are ranked according to their perceived importance. There seems to be an assumption that all the criteria are stationary—that they are assumed to be present on any specific project to some degree and do not change over time except potentially for the better with conscious effort. A little more formally, a process or system is said to be stationary if its behavioral description does not change over time, and nonstationary if its behavioral description does change over time.

TABLE. CRITERIA USED BY THE STANDISH GROUP TO GAUGE THE CHANCE OF PROJECT SUCCESS

Success Criteria
1. User Involvement
2. Executive Management Support
3. Clear Statement of Requirements
4. Proper Planning
5. Realistic Expectations
6. Smaller Project Milestones
7. Competent Staff
8. Ownership
9. Clear Vision & Objectives
10. Hardworking, Focused Staff

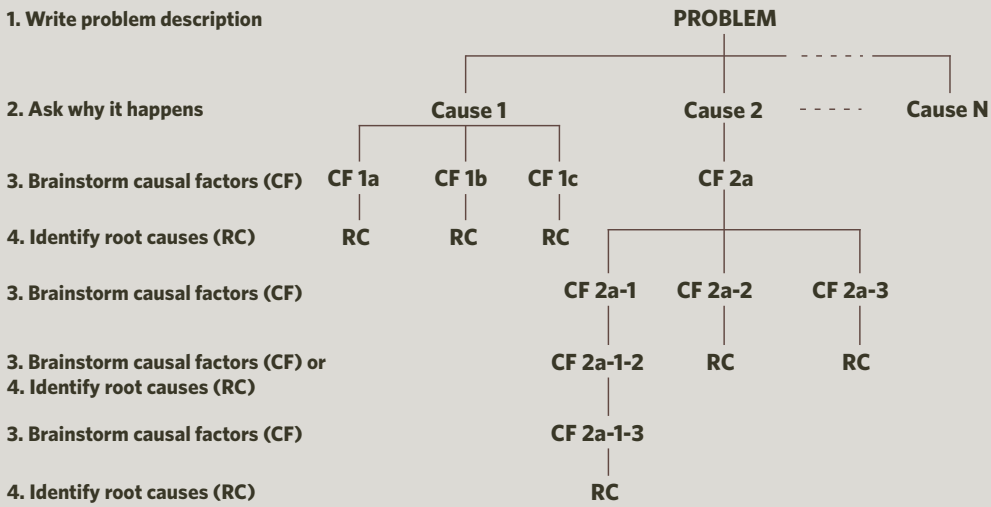
Systems under development exist in an environment that is not at all stationary over a project's development span. Technology changes in significant ways. Leaders retire or are replaced, and new leaders have new priorities and perceptions. New threats emerge and old threats diminish. Marketplaces shift as consumers change their buying habits in response to advertising and personal needs. Nonstationary environmental factors prevent requirements from being established early with the thought that they will not change. They will certainly change independent of the degree of discipline and process maturity on the part of the system developer.

The Five Whys

"A poorly defined problem and a rush to solution and action lead to activity without achieving the desired results" (Liker & Meier, 2006, p. 327). One recognized technique for defining problems and uncovering root causes of problems is to ask the *five whys*. Toyota refers to the five-whys process as a causal chain (Figure 1) because the questions and answers are chain-linked to help keep track of them. Perhaps the best way to explain the five-whys process for those not already familiar with the technique is to demonstrate it. The basic idea is to ask why about five times. The criteria from the Table suggest the causal factors that we can further explore to arrive at root causes of project failures.

So let's begin by defining the problem: *to discover why projects fail*. A possible first primary cause answer is: *because requirements change over*

FIGURE 1. CRITERIA USED BY THE STANDISH GROUP



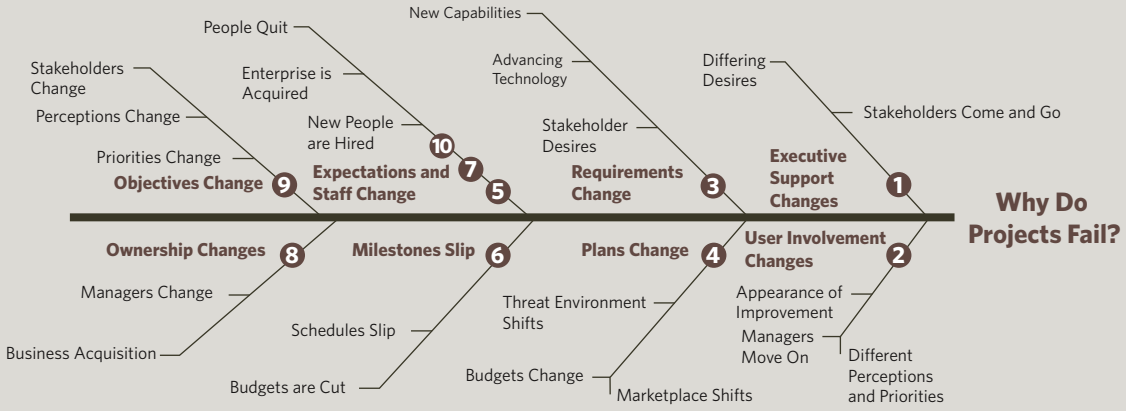
time. Then we seek causal factors with why No. 2: *Why do requirements change over time?* A possible answer is: *because advances in technology create opportunities.* Then we dig for deeper causal factors with why No. 3: *Why do advances in technology create opportunities?* A possible answer is: *because Moore's Law (1965) states that the number of components on a digital chip doubles every 18 months, which means digital products become practical that weren't practical earlier.* Then we dig again for deeper causal factors, with why No. 4: *Why do digital products become practical that weren't practical earlier?* A possible answer is: *because the complexity of software in the products increases to create new capabilities that demand more raw computing capacity and memory than earlier.* Then we seek the root cause with why No. 5: *Why does the increased complexity of software create new kinds of capabilities and create opportunities?* A possible answer is: *because stakeholders express a desire for new capabilities, and more complex software is the way to create them in the digital world in which we live.* When we ask good questions in the five-whys process and ask them of the right people, we quickly arrive at the root causes of problems.

We can further examine why projects fail by positing a second possible first cause: because *executive management support changes over time.* Then we seek causal factors with why No. 2: *Why does executive management support change over time?* A possible answer is: *because executive managers retire or relocate.* Then we dig for deeper causal factors with why No. 3: *Why does support change if executive managers retire or relocate?* A possible answer is: *because different managers have different priorities and perceptions*. Then we dig again for still deeper causal factors with why No. 4: *Why do different priorities and perceptions change support?* A possible answer is: *because executive managers have a vested interest in*

creating at least the appearance of improvements. Then we seek the root cause with why No. 5: *Why does improving things require different priorities and perceptions?* A possible answer might be: *because different priorities and perceptions provide the reason and justification for the improvements.* Again, we seem to have arrived at a root cause.

We can also diagram the root causes in an *Ishikawa diagram*, also called a *fishbone* diagram. Although further questions and answers are not detailed in this article, Figure 2 diagrams the results after asking the five whys for each of the 10 success criteria. Readers may wish to ask and answer the five whys to see if they achieve similar results.

FIGURE 2. AN ISHIKAWA OR "FISHBONE" DIAGRAM



The five whys and the Ishikawa diagram indicate that some—perhaps most of the root causes of project failures—are nonstationary. For example:

- A clear statement of requirements cannot be stationary because technology advances more quickly than ever, and marketplaces or threats in the environment shift.
- Executive management support and competent staffs must change in our world of international outsourcing and transient populations.
- Stakeholders' expectations cannot really be held constant over a project's life cycle regardless of whether or not they are realistic because stakeholders frequently change—not as a class, but as individuals.
- Ownership cannot remain constant in a marketplace of business resizing, reorganization, and acquisition.

Obviously, eliminating the nonstationary aspects of a project's environment is not practical. Is there any way to adapt to the changes? Though not comprehensively, at least three options can be independently adopted to mitigate the effects on a project in a changing environment. Before discussing the three options that are available, we will first review the historical setting of managing a project and review the evolutionary strategies that have recently been adopted.

Working in a Nonstationary Environment

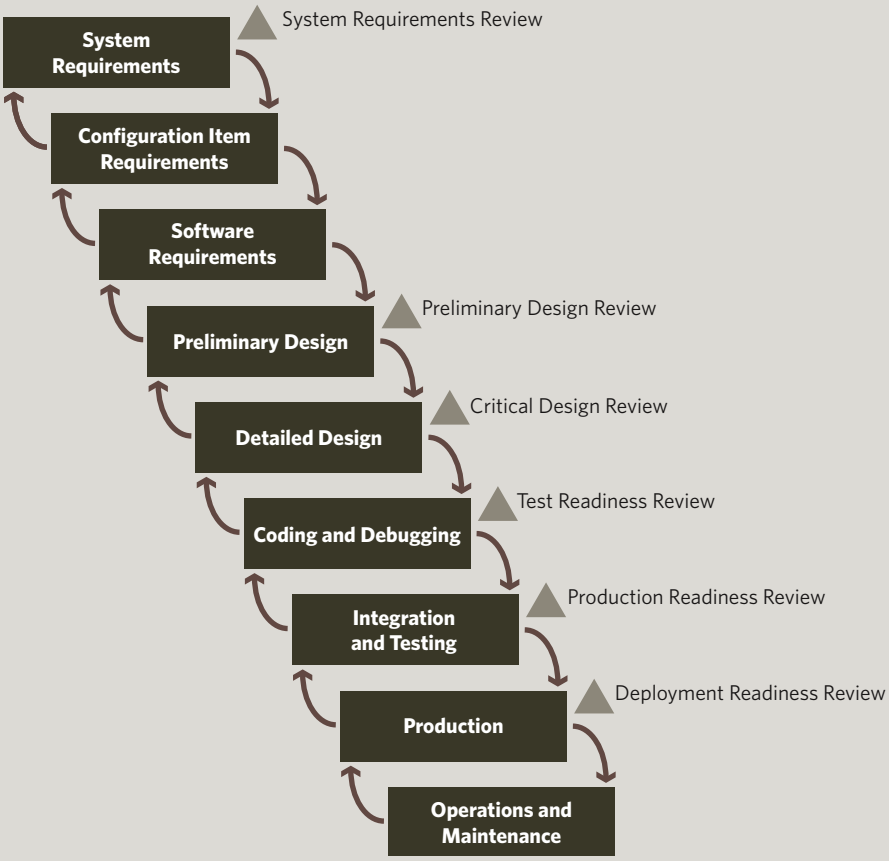
One historical method to acquire new systems is termed the *waterfall method* (Figure 3). Within the waterfall method, requirements are first established and then followed by several review milestones executed sequentially to arrive at a series of decisions that relate to the maturity of the system under design and development (Royce, 1970, pp. 1–9).

As is well known, the waterfall acquisition method can span a long time—perhaps years or even decades. The long span associated with the waterfall has been recognized as a factor in the failure of many projects that used it,[1] and this recognition led to alternative development methods such as the *spiral development method* defined by Barry Boehm (Boehm, 2002). Today there is recognition that systems evolve over their life cycles,[2] especially software systems, and the preferred approaches to system development are called evolutionary development (Pressman, 2001, pp. 34–47). Evolutionary development includes: (1) *incremental* development; (2) *spiral* development, including its win-win variations; (3) *concurrent* development; and (4) *component-based* development. For example, the Rational Unified Process (Larman, 2005) is a well-known, use-case-specified, architecture-driven, iterative software development process. The emphasis in these evolutionary development methods is on defining iterated shortened cycles that emphasize both risk reduction and increased product maturity in the subsequent repeated cycles. Evolutionary development leads to individual waterfall-like cycles that are individually short enough that the project environment is approximately stationary within the cycle.

Thus, the need to adapt to environmental changes is explicit. But this runs the risk of constant change, resulting in modification of requirements, objectives, visions, or support commitments at each cycle. Thinking of the environment as approximately constant for one cycle is not equivalent to imagining the environment is constant over the project's life cycle.

We could simply hold all requirements, visions, goals, plans, budgets, stakeholders, and staffs constant. We could view agreed-to plans as commitments, but then the risk is that we will develop systems or capabilities that are not congruent with the marketplace or threat environment; and

FIGURE 3. THE WATERFALL METHOD



that would be like Ford creating an Edsel without paying attention to what customers want. Is there anything else we could do?

Mitigating the Effects of Nonstationarities

In a competitive setting, any project must address cost, performance, marketing features, technical maturity, and time to completion. Yet, *nonstationary environments* imply acquisition projects will continue to experience product configuration changes and other changes that drive up cost and extend schedules. We should deal with changes in a sensible way. Being sensible is tantamount to adopting heuristics[3] to deal with environmental changes. And, what is sensible depends on what we consider to be the most important variables to control. The priority given to cost and schedule will vary product to product, market to market, and threat to threat.

Given the fact that we cannot eliminate the nonstationary aspects of a project's environment, at least three options are available to mitigate

the effects of the nonstationarities: (1) control cost, (2) control schedule, and (3) manage changes with discipline. We could give highest priority to cost and try to control cost to avoid the nonstationary effects on cost from the environment. We could just as well constrain schedule to avoid the nonstationary effects on the schedule from the environment. And we must manage the changes in a disciplined way to avoid the worst effects of the nonstationarities.

DESIGN TO COST

The *Innovator's Dilemma* (Christensen, 1997) makes it clear why cost continuously becomes an important factor that affects the competitive position of commercial companies. And newspapers and television newscasters regularly remind us that the cost of defense acquisitions by the U.S. Government repeatedly surfaces as an area of concern to the Congress and taxpayers. When cost is the most important variable, yet a constraint is defined that cost cannot exceed a preset limit, then we are dealing with a *design-to-cost* paradigm.

A design-to-cost strategy aims to control costs by treating cost as an independent design parameter. A substantial fraction (70 to 80 percent) of a product's cost is determined during the product's design/development phase. According to Crow (2000), the elements of a design-to-cost approach include the following:

- Recognition of what the customer can afford
- Definition and allocation of the target costs to a level at which costs can be effectively managed
- Commitment on the part of designers and development personnel
- Stable management to prevent requirements creep
- Understanding of cost drivers and their management in establishing product specifications
- Early use of cost models to project design/development costs in support of decision making
- Active consideration of costs appropriately weighted during development
- Exploration of the product's trade space to find lower cost alternatives
- Access to a database of past costs to provide quantitative information about present cost estimates
- Design for manufacturability and design for assembly to avoid rework and its associated costs
- Identification of functions that have a high cost-to-function ratio as targets for cost reduction

- Consistent cost accounting methods, models, and processes
- Continuous improvement through value engineering to improve products' value over time.

Thus, well-understood techniques and practices are readily available that treat cost as an appropriately weighted design parameter. Adopting these techniques when cost is a high priority can serve to limit costs and thereby reduce the impact of cost growth due to nonstationary environments.

DESIGN TO SCHEDULE

In a military setting, quick reaction implies that a capability is required in the field with high priority in a time period that can be as short as 1 to 3 months. Environments do not change appreciably in that timeframe. Time to market can also be a consideration for commercial firms because of short windows of opportunity. If schedule is the most important variable, and a constraint is defined that project length cannot exceed a relatively short preset span, we are dealing with a *design to schedule* paradigm, which is often also called a *Quick-Reaction Capability* (QRC) paradigm.

We cannot find much written about QRC other than definitions of the abbreviation. But our private industry experience gives us some personal insight into how to accomplish a QRC effort. Basically, a QRC effort relies on the reuse of earlier designs and components, and upon a dedicated, knowledgeable workforce that is committed to completion of the effort in the required timeframe. The reuse of standard parts eliminates the long lead time to design new or nonstandard parts. The reuse of standard manufacturing processes and tools eliminates time to retool or re-plan the manufacture. And techniques that are suggested to implement the *design-to-cost* paradigm suggest further techniques to save time: time correlates with cost. We can reword the recommended elements of the design-to-cost paradigm (cited above) to apply to the QRC paradigm:

- Recognition of when the customer needs the product or capability
- Definition and allocation of schedule milestones to a level at which time can be effectively managed
- Commitment on the part of designers and development personnel
- Stable management to prevent requirements creep
- Understanding of time drivers and their management in establishing product specifications
- Early use of schedule models to project design/development time in support of decision making
- Active consideration of time appropriately weighted during development

- Exploration of the product's trade space to find lower elapsed-time alternatives
- Access to a past experience database to define earned-value milestones realistically
- Design for manufacturability and design for assembly to avoid rework and its associated time
- Identification of components and parts that have a high time-to-capability ratio as targets for schedule reduction
- Consistent earned-value milestone accounting methods and processes to assess technical progress
- Continuous improvement through value engineering to improve the time to market/field.

Thus, inferred techniques and practices are available that treat schedule as a constrained parameter. Adopting these techniques when time is a high priority can serve to limit schedule and thereby reduce the impact of schedule growth due to nonstationary environments.

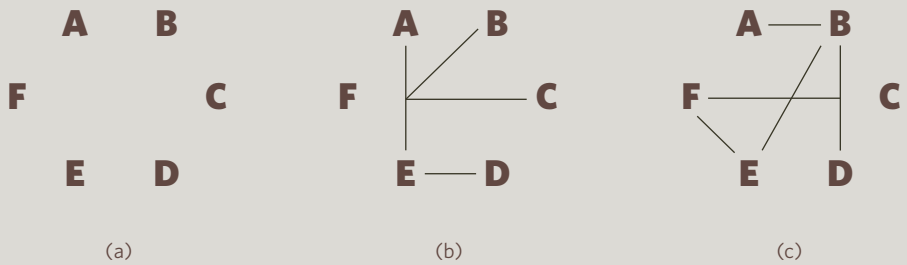
CONFIGURATION AND CHANGE MANAGEMENT

This article is not deliberately focused on a single weapon system or product, which is likely easier to change. Modern acquisitions include systems of systems in an environment of exponentially increasing inter- and intra-dependencies. In an era of net-centric warfare, globalization, and the World Wide Web, interdependencies are unavoidable. In his book *Leading Change*, Kotter (1996, pp. 21, 136–137) discusses the nature of change in highly interdependent systems. Specifically, in highly interdependent environments, a single desired change drives almost everything to change (Figure 4). We think of physical changes to a system as *configuration* changes. But, interdependency becomes a further challenge when various component systems are themselves unstable, for example, because of funding constraints, political climate, or changes in leadership or ownership. Therefore, *change management* deals with nonphysical aspects of a system, such as requirements changes, priority or budget changes, or other changes to established baselines.

Kotter (1996) also highlights an eight-step process of creating major change:

1. Establish a sense of urgency.
2. Create the guiding coalition.
3. Develop a vision and strategy.
4. Communicate the new vision.
5. Empower broad-based action.
6. Generate short-term wins.

FIGURE 4. CONFIGURATION MANAGEMENT USES THE SAME BASIC PROCESS THAT CHANGE MANAGEMENT USES



Note. (a) In a system with independent parts, A can be changed by simply changing A; (b) in a system with some interdependence, several elements (A, E, D) may need to be changed in order to change A; (c) in a system with much interdependence, all elements may need to be changed in order to change A.
(Blanchard & Fabrycky, 2006)

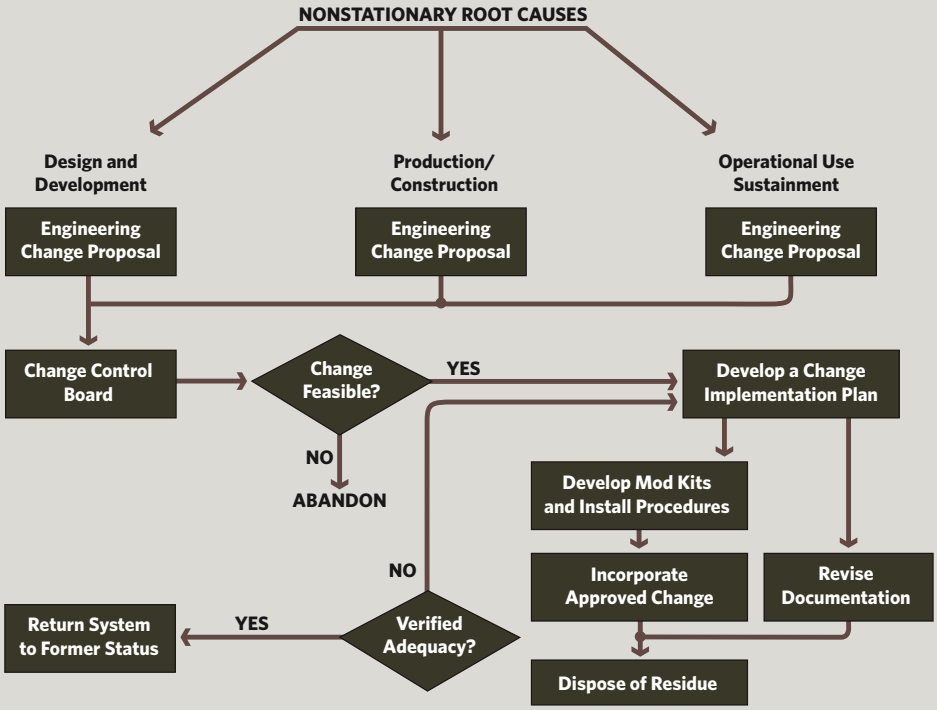
7. Consolidate gains and produce more change.
8. Anchor new approaches in the culture.

While Kotter's eight-step process model is primarily designed for changing organizations, many of the principles contained in this approach are applicable to effective project/program management, especially during periods of frequent change and turmoil. This is especially true if the desired change is more process- than product-based. For example, if a project desires to capitalize on a new technology or address quality problems through manufacturing process alteration, we can see where the Kotter eight-step process is directly applicable.

Configuration management. Change management and configuration management are closely related (Figure 5). The concepts are similar: by controlling or managing proposed changes to a product's or a system's configuration, we are controlling or managing the effect on the product or system of changes in the external environment. Configuration management typically requires a configuration control board chaired by an executive stakeholder; a configuration working group of subject matter experts to analyze proposed changes, and to create and evaluate alternative means to accommodate the change; and a secretariat to record the deliberations and decisions of the configuration control board, and to manage any action items that are assigned.

Change management. Change management is a well-known and respected means to deal with volatile requirements, budget cuts, and other nonstationary root causes of project failures. The definition of change

FIGURE 5. BASIC PROCESS FOR CONFIGURATION MANAGEMENT AND CHANGE MANAGEMENT



Note. Adapted from Figure 5.10 (Blanchard & Fabrycky, 2006, p. 138).

management includes at least four basic aspects: (1) the task of managing change, (2) an area of professional practice, (3) a body of knowledge, and (4) a control mechanism. Change can be planned or unexpectedly driven by unforeseen external events; this article addresses the latter. Many believe that the general process of change can be treated separately from the specifics of the situation; thus, acquisition practitioners may seek to leverage the expertise of professional change consultants.

The body of knowledge relating to change management is drawn from psychology, sociology, business administration, economics, industrial engineering, systems engineering, and the study of human and organizational behavior. For many practitioners, these component bodies of knowledge are linked and integrated by a set of concepts and principles known as general systems theory (Skyttner, 2005). Thus, a large, somewhat eclectic body of knowledge underlies the practice of change management upon which many practitioners might agree.

However, the very application of the word *management* in direct association with the word *change* implies that change is an activity or event that lends itself to being controlled (control is a function of management)

through the application of logical procedures applicable to standardized, effective, and efficient processes. For this to be true, the volatility of impacting factors must also have some reasonable degree of predictable control. Infrequently, however, is there a case where change is desired while all aspects of the change are predictable. Therefore, although various methods are available by which one can approach change, the challenge of successfully effecting change is directly proportional to the number of nonpredictable aspects.

Summary and Conclusions

Cobb's Paradox, as detailed in this article, is a result of nonstationary causes. Some of these nonstationary causes are not explicitly defined in the acquisition or systems engineering literature before now. Some of these nonstationary causes are not at all easy to manage. So it seems that project failure is not paradoxical as Cobb's Paradox suggests.

Given the fact that project environments can not be expected to remain constant over a typical project's life cycle, we are left with disciplined change management to deal with any changes and heuristic methods to control their impacts. Perhaps if we treat every project as if it was both cost-constrained and schedule-constrained, and we applied disciplined change management techniques, we would avoid many of the project problems analyzed by The Standish Group and so clearly articulated by Martin Cobb.

Author Biographies

Lt Col Joseph W. Carl, USAF (Ret.) enlisted in the U.S. Air Force at age 17 and retired after a 25-year military career. Following his military service, Dr. Carl worked for Harris Corporation as a systems engineer for over 21 years, and for Riverside Research Incorporated as an adjunct faculty member of the Air Force Institute of Technology (AFIT) where he taught systems engineering. He holds a PhD from Ohio State University and is a Professional Engineer (PE) and Certified Systems Engineering Professional (CSEP).

(E-mail: Joseph.Carl.ctr@afit.edu)

Col G. Richard Freeman, USAFR (Ret.) is the technical director, Air Force Center for Systems Engineering, AFIT. He has over 30 years' experience in systems and process engineering. Col Freeman's federal civilian career included positions as chief, Concept Development and Process Engineering; chief process officer; and chief, Environment, Safety and Occupational Health for Weapons Systems. In the defense industry, he held executive positions as CEO Delta Environmental Services, Inc.; executive vice president and board member, EICON, Inc.; and positions with General Electric, United Nuclear, and UNC Aerospace. Col Freeman holds a BS from Phillips University, cum laude, an MA from the National War College with highest distinction, and an MS from Troy Tate University, summa cum laude, respectively. His professional credentials include CSEP and CSEP-Acquisition.

(E-mail: Richard.Freeman@afit.edu)

REFERENCES

Blanchard, B. S., & Fabrycky, W. J. (2006). *Systems engineering and analysis* (4th ed.). Upper Saddle River, NJ: Pearson-Prentice Hall.

Boehm, B. W. (2002). *Spiral development: Experience, principles, and refinements.* Pittsburgh: Carnegie Mellon University, Software Engineering Institute.

Christensen, C. M. (1997). *The innovator's dilemma: When new technologies cause great firms to fail.* Oxford, MA: Harvard Business School Press.

Cobb, M. (1995). *Unfinished voyages.* Presentation at The CHAOS University, sponsored by The Standish Group, in Chatham, MA, November 6–9, 1995.

Crow, K. (2000). *Achieving target cost/Design-to-cost objectives.* Retrieved from http://www.npd-solutions.com/dtc.html

Kotter, J. P. (1996). *Leading change.* Oxford. MA: Harvard Business School Press.

Larman, C. (2005). *Applying UML and patterns: An introduction to object-oriented analysis and design and iterative development* (3rd ed.). Upper Saddle River, NJ: Prentice Hall PTR.

Liker, J. K., & Meier, D. (2006). *The Toyota way fieldbook: A practical guide for implementing Toyota's 4Ps.* New York: McGraw-Hill.

Moore, G. (1965, April 19). Cramming more components onto integrated circuits. *Electronics Magazine, 38*(8).

Pressman, R. S. (2001). *Software engineering: A practitioner's approach.* New York: McGraw-Hill.

Royce, W. R. (1970, August). *Managing the development of large software systems.* Proceedings of the IEEE/WESCON Conference, Los Angeles, CA.

Skyttner, L. (2005). *General systems theory* (2nd ed.). Singapore: World Scientific Publishing Co.

U.S. Government Accountability Office. (2006). *Defense acquisitions: Assessments of selected major weapon programs* (GAO-06-391). Washington, DC: Author.

Young, J. (2007). *Prototyping and competition* [policy memorandum]. Washington, DC: Office of the Under Secretary of Defense (Acquisition, Technology & Logistics).

ENDNOTES

1. According to The Standish Group, 16 percent of software-intensive projects are successful, while 53 percent are over schedule or budget and 31 percent are cancelled (see http://www.gtislig.org/Documents/ISO%2012207.ppt#265,10,Project Failure Reasons).
2. Note that evolution is driven by feedback from the environment, and is usually interpreted to result in entities that have adapted to the changing environment to become more survivable in it.
3. A heuristic is something that cannot be proven to work all the time, but experience indicates it works well most of the time. A heuristic may also be thought of as a method of solving a problem for which no formula exists so that the solution is based on informal methods or experience and may employ a form of trial-and-error iteration.