SYM-AM-16-061

PROCEEDINGS
OF THE
THIRTEENTH ANNUAL ACQUISITION RESEARCH SYMPOSIUM

THURSDAY SESSIONS
VOLUME II

Acquisition Program Teamwork and Performance Seen Anew: Exposing the Interplay of Architecture and Behaviors in Complex Defense Programs

Eric Rebentisch, Research Associate, MIT
Bryan Moser, Lecturer, MIT
John Dickmann, Vice President, Sonalysts Inc.

Published April 30, 2016

Prepared for the Naval Postgraduate School, Monterey, CA 93943.

ACQUISITION RESEARCH PROGRAM
GRADUATE SCHOOL OF BUSINESS & PUBLIC POLICY
NAVAL POSTGRADUATE SCHOOL

The research presented in this report was supported by the Acquisition Research Program of the Graduate School of Business & Public Policy at the Naval Postgraduate School.

To request defense acquisition research, to become a research sponsor, or to print additional copies of reports, please contact any of the staff listed on the Acquisition Research Program website (www.acquisitionresearch.net).

ACQUISITION RESEARCH PROGRAM
GRADUATE SCHOOL OF BUSINESS & PUBLIC POLICY
NAVAL POSTGRADUATE SCHOOL

Panel 16. Improving Governance of Complex Systems Acquisition

Thursday, May 5, 2016	
11:15 a.m. – 12:45 p.m.	**Chair: Rear Admiral David Gale, USN,** Program Executive Officer, SHIPS ***Complex System Governance for Acquisition*** Joseph Bradley, President, Leading Change, LLC Polinpapilinho Katina, Postdoctoral Researcher, Old Dominion University Charles Keating, Professor, Old Dominion University ***Acquisition Program Teamwork and Performance Seen Anew: Exposing the Interplay of Architecture and Behaviors in Complex Defense Programs*** Eric Rebentisch, Research Associate, MIT Bryan Moser, Lecturer, MIT John Dickmann, Vice President, Sonalysts Inc. ***A Complex Systems Perspective of Risk Mitigation and Modeling in Development and Acquisition Programs*** Roshanak Rose Nilchiani, Associate Professor, Stevens Institute of Technology Antonio Pugliese, PhD Student, Stevens Institute of Technology

Acquisition Program Teamwork and Performance Seen Anew: Exposing the Interplay of Architecture and Behaviors in Complex Defense Programs

Eric Rebentisch—is a Research Associate at MIT's Sociotechnical Systems Research Center. He leads the Center's Consortium for Engineering Program Excellence and previously led the Lean Advancement Initiative's Enterprise Product Development research. Dr. Rebentisch's portfolio of research projects includes studies of the integration of program management and systems engineering and performance benchmarking of the U.S. shipbuilding industry. [erebenti@mit.edu]

Bryan Moser—is the Founder and CEO of Global Project Design, Inc. (GPD), and Lecturer in the Systems Design and Management program at MIT. His industry engagements in the U.S., Europe, and Japan have provided a laboratory for development of *project design* workshop tools and methods. Prior to GPD, for a decade with United Technologies Corporation, Dr. Moser led research on and implementation of global technology and product systems development. He has published extensively on dispersed teamwork and the interplay of coordination architectures and team behaviors impacting complex project cost, schedule, and performance. [bry@mit.edu]

John Dickmann—is a Vice President at Sonalysts Inc. He served for 22 years as an active duty Navy submarine officer. He is a graduate of MIT's Engineering Systems Division and has researched lean processes and enterprise design for the U.S. Air Force, the Naval Sea Systems Command (PMS-401), and the Office of the Secretary of Defense (Net Assessment). Dr. Dickmann manages systems design projects and conducts operational and strategic technology analysis for defense and industry clients. [jdickmann@sonalysts.com]

Abstract

This research frames complex engineering development programs as sociotechnical systems with program performance driven by interpersonal and inter-organizational dynamics as well as technical system interdependencies. It attempts to address the question of why performance in complex development programs has not improved significantly in the last several decades, despite the development and application of many new and sophisticated tools for managing these programs. A review of the literature on managing complex sociotechnical systems was used to develop a framework and method for instrumenting complex engineering programs and measuring their essential attributes. The proposed framework identifies fundamental elements of engineering programs (relating to, e.g., products, processes, organizations, and people) and the drivers of program performance. The framework is illustrated using a case study of a complex engineering program that spanned multiple technical systems, organizations, and disciplines. The paper discusses the resulting measurement framework and provides examples of the application of the framework to identify management control "levers" for design, engineering, test and evaluation, fielding, and sustainment of complex engineering programs.

Introduction

Large-scale engineering programs are challenging to complete within planned parameters. U.S. Department of Defense (DoD) programs involve outlays of public finds, and are therefore well-documented. They unfortunately often report disappointing outcomes. The U.S. Government Accountability Office (GAO) reported in 2009 that the accumulated cost overrun of the largest 96 DoD engineering development programs reached nearly $300 billion, with an average schedule overrun close to two years (GAO, 2009). This doesn't appear to be an aberration from the early part of the 21st century. The GAO reported previously that combined cost overruns for large development programs (programs totaling more than $1 billion for research, development, testing and evaluation in fiscal year 2005 dollars) initiated in the 1970s exceeded the DoD's initial investment estimate by 30%, or $13

billion (in fiscal year 2005 dollars), with equivalent overruns of 39% in the 1980s and 40% in the 1990s (GAO, 2006). Despite numerous acquisition reform efforts and policy revisions during those years, defense acquisition programs during that three decade period continued to routinely experience cost overruns, schedule slips, and performance shortfalls. Underperformance is not confined just to defense development programs, though. Reports of disappointing performance in large-scale civil engineering programs tell a similar story (Cantarelli et al., 2010).

Poor development program performance seen across a number of different applications and business sectors suggests that there may be underlying, systematic factors that bias programs toward trouble. An example from commercial aerospace illustrates just such a diverse array of challenges. In 2003, Boeing launched the 7E7 program as a refresh and partial replacement for its 767 and 747 families of aircraft. The 7E7 eventually came to be known as the 787 and quickly established a very strong order book from airlines ("Boeing 787 Dreamliner," 2016). By the time it finished development, the program exceeded the estimated costs by three times and took roughly twice as long to develop as estimated. An investigation into the reasons for these outcomes suggested that a likely cause was an excessive growth of development project complexity. The aircraft itself became more complex due to the use of new materials which was in many cases beyond the capacity and experience of Boeing or its suppliers. The organization tasked with developing this new aircraft also became more complex because of significantly increased external development through partners and vendors, outsourced system integration and higher interdependence through a more parallelized development process (Allworth, n.d.). Insufficient ability to handle growth in both types of complexities eventually led to the major project delay and skyrocketing costs, pushing the program into crisis (Denning, n.d.). Although the 787 eventually overcame the crisis, each delivered 787 still generates losses (Gates, 2016). The development of the Boeing 787 aircraft design and production required coordination across the globe of multiple organizations, corporations, and governments, which is representative of many development projects, where estimated costs and development are exceeded.

The same challenges plague complex programs in public sector layered infrastructure initiatives and commercial services deployment. Current program management standards are largely heuristic, experience-based, and generic, with application oriented toward a wide range of sectors, project scope, and activities. They are, however, deficient in addressing the management of interdependencies and cross-boundary interactions in complex programs such as those seen in the 787 development program. These interdependencies and interactions include those that are hierarchical between management layers, lateral between functional groups, and multi-scale based on nested layers of performance domains. While hierarchical and lateral interdependencies are acknowledged in traditional organizational/management literature, they are seldom defined at the level of specificity that is required for program management. Multi-scale relationships are increasingly pervasive in operations, driven by increasing complexity in engineered systems, but they remain largely undefined at a useful level of specificity in either the organizational/management or program management literatures.

The lack of improvement in complex program outcomes may be traced to few new and more effective program management practices and, ultimately, too little innovation in the way that the basic attributes of programs are measured. Prevalent project management standards rest upon heuristic practices. Systems engineering standards address system architecture, methods, and tools, but fall short in defining social and managerial interdependencies with a product system and its design and operating context. Identifying and articulating measurement and instrumentation for program elements at a fundamental

level could lead to a richer characterization of the essential attributes of complex programs. The resulting richer datasets might then enable the development of new analytic methods for understanding the underlying drivers of program performance. The framework explained in this paper is part of an effort to move program management from its primary basis in heuristics and collections of best practices toward the design of projects based on a deeper understanding of the interdependencies, behaviors, and performance of sociotechnical networks under systemic complexity.

Current Measurement and Control Systems for Complex Programs

Relatively simple programs (i.e., linear extrapolations from known space to known space, perhaps best characterized by manufacturing), while potentially complicated, may not be particularly complex. That is to say, they may have many elements, but they and the relationships between them are all relatively well-understood and predictable in their behaviors. The associations between tasks, participants, and sequencing may be well-specified, and the imperative is to execute the tasks through clearly-defined relationships. In these types of programs, traditional tools, practices, and methods for managing the program may be entirely adequate to this challenge.

However, when novelty, scarcity, or uncertainty are part of the work space, there are potentially more known unknowns or even unknown unknowns than would be expected in a simple program. This may require learning, innovation, and possibly improvisation from the program team to deal with exceptions to the plan or other unexpected developments. With these emergent behaviors, a linear extrapolation from known practices will be of limited benefit. Learning will require the flow of information from a range of different sources in order to notice, acquire, understand, and synthesize knowledge into needed new forms. This suggests exploration and exploitation of the information (and related resource) networks across the program. This would not be possible without understanding and controlling the dependencies in the system.

Evolution of Systems Projects Control

We can trace the underlying model of work used most commonly in project management back to Taylor and Gantt (Gantt, 1903; Wilson, 2003), with jobs described as sets of discrete tasks, refined over time to reduce variation in repeatable activities of fixed duration and fixed sequence. Both Taylor and Gantt (1903) were managing factories, with the Gantt chart itself originally a table to describe fixed jobs for workers. These same assumptions of standardized durations and sequence were carried forward in the middle of the last century with the advent of critical path (CPM; Kelley & Walker, 1959) and other network techniques.

The complexity of government programs and a shift towards cost control led in that next decade to a more centrally-controlled project management approach. The roots of today's project control, including earned value, emerged in the 1960s in requirements for defined work breakdown structures (WBS), systems engineering management plans (499), and Cost/Schedule Control System Criteria (C/SCSC). Still, the underling view of task as project "atom" connected through a precedence based network remained.

Amidst a call for simplification compared to prior heavy processes, defense acquisition management was transformed starting in 1991 through the DoD Directive 5000 series (Dillard, 2003). Over the next decade, in parallel with streamlining through adoption of commercial practices in government programs, improved risk management emerged, including attention to technology readiness. The underlying model of project work spread through the introduction of CMMI by SEI and other standards, including PMBOK in North

America. Like the century earlier factory models of work and the mid-century defense approaches, these followed the same underlying model of project work as tasks connected by precedence networks of dependencies.

The emergence of project control beyond large government programs can be partly traced to Fleming and Koppelman, who promoted earned value over two decades as Koppelman built Primavera Systems (now part of Oracle). They commented on the potential of earned value as a concept not only in large defense programs but—in simplified form—for software and other projects (Fleming & Koppelman, 1994, 1996). Their papers and book in the PMI community from the middle to late 1990s introduced earned value to a broader project management audience in industry.

Earned Value

Earned Value management is well-documented by many; this paper assumes the reader has already or can easily gain EVMS basics, so it does not include reiteration of the earned value approach. Instead, we trace the evolution of EVMS from its emergence from the DoD in the early 1990s to recent years.

The A-12 program cancellation in 1991 has been widely accredited to early indicators of cost variation from EVM. However, Christensen's (1994) work showed, too, that after the first third of the program, the cost performance index will stabilize, with limited variability as a project proceeds. Still, as an early indicator of the A-12 Program cost overruns, the method performed as expected (Christensen, 1994).

Evolution of Earned Value: ES and ED

Earned Schedule

In 2003, Walter Lipke, at the time head of the software division in the U.S. Air Force's Oklahoma Air Logistics Center, proposed a change to the schedule measures in EV (SV–Schedule Variances and SPI—Schedule Performance Index) that he called Earned Schedule (ES; Lipke, 2003, 2004). Lipke and others at the time had noticed that the schedule variance (SV and SPI) as used in earned value became less predictive later in a program. Since EV schedule measures are derived from cost (BCWP–Budgeted Cost for Work Performed, and BCWS–Budgeted Cost for Work Scheduled), as an over-schedule project approaches completion, the schedule variance according to SPI approaches 1.0. and SV approaches $0. Instead, in Earned Schedule (ES), as a substitute for budgeted cost as performed versus as scheduled (BCWP and BCWS), Lipke takes the actual duration versus the planned duration for the work performed. Lipke was able to show that in some cases, the earned schedule measures remain meaningful in the latter stages of a program, including showing positive or negative variance at completion if the actual duration differed from the planned duration.

While ES is an improvement over EV and can be implemented using the existing EV metrics, ES still rests upon schedule progress as derived from a portion of original budget spending, with a linear association to schedule progress.

Earned Duration

In a recent and important contribution, Khamooshi and Golafshani (2014) take a step further than Lipke's Earned Schedule approach. They refer to their methods as "Earned Duration" (Khamooshi & Golafshani, 2014). Consistent with Lipke's response to the inaccuracy of EVM schedule-related performance, they propose complete decoupling of the schedule metrics and forecasts from cost related inputs.

The authors point out that as patterns of cost over time, progress over time, and value of scope over time become non-linear, then the assumptions of uniform linear association between spending, progress, and likelihood of ultimate schedule performance become false. Instead, they define earned duration as “Earned Duration of scheduled activity i: EDi, at any point in time, is the value of work performed expressed as proportion of the approved duration assigned to that work for activity (e.g., days)” (Khamooshi & Golafshani, 2014).

In their paper, they also, interestingly, emphasize the dual role of project control techniques: first to ascertain the performance of a project to date as compared to some original baseline plan, and second to determine the accuracy of original estimates, providing a view that will allow comparison and learning across multiple projects.

While the data necessary to calculate earned duration ("ED(t)") goes beyond that required by classic EVM, they argue that the data is available otherwise, as was necessary for original planning and ongoing scheduling by project teams. Importantly, the measures require an estimate of remaining duration, given the current state of the project, as an indicator of progress rather than cost. Whether these estimates are made at the macro or micro level is a project control or architectural decision irrespective of which measures are chosen.

The authors also point out that use of actual performance metrics from an earlier stage of a project as a proxy for expected performance in later stages, given the typically limited information about specific resources, priorities, and other externalities, is “questionable”: “If the stages of the project are different and heterogeneous, which normally is the case, there is no rationale for assuming past performance is a good predictor of the future” (Khamooshi & Golafshani, 2014).

EV Variants and Control Points

Colin and Vanhoucke survey recent literature on earned value and project control techniques, characterizing the set from the original use of Critical Path Method (CPM) from Kelley and Walker (1959) as a bottom-up approach and the more recent earned value and its variants which are described as top-down. They assert that these methods vary in the number and position of control points—positions in a WBS at which are placed monitors for observing and buffers for controlling project flow. EVM as practiced rests upon a topmost WBS element with calculated EV, PV, cost and schedule metrics; they discuss several recent papers which show control points at key points in the project, not necessarily the complete data collected bottom up from each WBS activity. For example, another method by Lipke (2012) places control points along the critical path. They introduce two approaches inspired by Goldratt's Critical Chain method (Goldratt, 1997), in which they explore control along the critical path, along feeding paths into the CP, or instead entire subnetworks which feed the CP.

Their approach recognizes that a program manager (the PM) is burdened by the amount and upkeep of control data and response. By seeking a balance between bottom-up and top-down approaches, they seek to minimize overzealous control that—being unsustainable—causes latent and poor quality control signals. However, their paper does not directly address resource capacities, and only indirectly the capacity of a PM to handle control activities.

Furthermore, as the various methods in Colin and Vanhoucke, Lipke and others rest upon an underlying model of project as network with discrete precedence dependencies, the capacity and quality of interactions across these project interfaces is misrepresented. In selecting the positions of control points, one must ask, “Who calculates, interprets, and

reports the information at that interface?" Are their experience, capacity, and accountability aligned with the demanded attention, in timely fashion, to the control point in balance of all the other demands "on their plate"?

Our thinking, when viewing complex systems projects as sociotechnical systems, places an emphasis not only on the PM, but on distributed resources, their condition of awareness and attention, their attempts to work and interact, who make mistakes and correct them, and learn over time. Therefore, not only the PM, but those project participants who would be best positioned to own control points and their source knowledge, should be considered. By analogy, if a control system lacks the capacity to process input signals in real time, it will become saturated and lose its control authority. The capacity of the system can be increased by parallel processing using a system of distributed controllers. However, that approach only works if the distributed controllers are coordinated and the interdependencies between them are managed. We assert that a burden of demands to be aware of and interact for project governance be distributed to align both with capacity and inherent capabilities of resources.

Summary

The legacy of program management and systems engineering tools and methods is rooted in practices suited to factory operations where tasks are assumed to be well-defined, dependencies between tasks are relatively simple, and the flow of work is fairly linear. These assumptions have changed little since a century ago, or at least have not been challenged in a vigorous way. Yet, it is clear that the complexity, both static and dynamic, of development programs and their corresponding sociotechnical systems has increased significantly in both scale and nature over the same time period. As a consequence, it should not be surprising that complex developments suffer poor outcomes and are seemingly uncontrollable. Their program management control systems have not kept pace with their changing nature. The case study introduced in the next section illustrates some of the ways in which management control systems may be challenged in a complex program.

Re-Architecting the Submarine Sonar Sociotechnical System

Our case study examples highlight the challenges of getting people to act with awareness and conviction at critical interfaces. We find that these are especially difficult when change causes patterns of demanded coordination to shift from historical ones. Whether due to habit, old incentives, or gaming—shifts in behavior are necessary to improve both local and systemic performance. This is especially true when program managers are confronted with shifting externalities such as changing technologies, shifting operational performance requirements and fiscal constraints. Successful program leaders understand the need to shift organization and system architectural alignment in these conditions.

The case of Navy submarine sonar system program management in the 1990s is an example of these types of change and the consequent need to recognize and design changed dependencies. To date, many published lessons drawn from this case fall in the category of best practices. How can we move from heuristics to a repeatable, measurable approach that can provide program managers real or near-real time feedback on internal program coordination demands?

Crisis: Loss of Submarine Acoustic Superiority

Military competition in the undersea consists of a constant evolution of operations and technologies to detect and to minimize acoustic emissions. In the early 1990s, U.S. submarines lost their long-standing acoustic advantage against Russian submarines—the ability to detect and track them before detection by them.

Throughout the Cold War, the Navy had invested billions of dollars in acoustic research and advanced sonar systems, concentrating its effort on custom-designed digital signal processing systems with military-unique components and tightly integrated proprietary hardware and software. Specifications were developed by the Navy Laboratory, and detailed design and production of hardware and software were conducted at one of two prime contractors. System development and fielding took a decade or more from conception and cost billions of dollars, with unit costs ranging to hundreds of millions of dollars.

The loss of acoustic superiority coupled with post–Cold War budget cuts created pressure to improve the performance of U.S. submarine sonar systems quickly. Added to this pressure was a wider questioning of the relevance of submarines in the "new world order," which created an organizational crisis, spurring what is arguably the most successful acquisition reform effort in U.S. defense industry history.

Diagnosis and Prescription

The Submarine Force response to this technical and fiscal challenge was to examine the problem from a fact-based perspective. In early 1995, the Submarine Superiority Technology Panel (SSTP), a panel of acoustics experts, was established to examine the technical performance of submarine sonars. It came to two major conclusions: (1) The legacy sonar system technical architecture was ill-suited to leverage Moore's Law for increased signal processing power and (2) the system development and acquisition organizations and the development-acquisition process inhibited experimentation with new algorithms.

The panel observed that there was no viable means to test, evaluate, and integrate advanced algorithms that had been developed by academic researchers through the 1980s. They recommended a commercial-off-the-shelf (COTS) architecture and an "open" development process. By the fall of 1995, the Submarine Superiority Management Council (SSMC) was established to address the findings of the SSTP. The SSMC worked through the spring of 1996, by which time, the core problems of boosting processing power and developing a means to inject new ideas into the sonar system were the main focus of attention. From this time onward, the responsible program managers for submarine sonar systems were working on developing new technical, process, and organization architectures for the development and acquisition of advanced technology for submarine sonar systems.

System Solution

The program managers charged with submarine sonar embarked on an effort to identify viable commercial processing technologies, identify and develop improved signal processing algorithms, and to develop a process to field these improvements quickly. This collective effort was reflected in new technical and organizational architectures—new dependencies and interfaces among the organizations involved in the submarine sonar program and in the functional implementation of the sonar system.

Architecture Changes

Architecture is the overall scheme by which the functional elements of a system (technical, people, organizational) are partitioned to individual subsystems/teams and are arranged with respect to each other. Architecture sets the rules which govern interactions in

and among systems, both in *operation and over time*, as they evolve in response to changing technology, operational demands and external constraints (Ulrich & Eppinger, 1995; Moses, 2006; Henderson & Clark, 1990; Garlan & Shaw, 1993; Clark et al., 2004; Clark et al., 2005; Board, 2000).

Change the System Technical Architecture

The major technical change was to separate hardware and software into a layered architecture. This was achieved by the development of middleware, a set of software that served as an interface between commercial processors and proprietary Navy algorithms. This change enabled the program managers to leverage the cost and processing benefits of Moore's Law without the need to change the existing software.

Change the Development and Acquisition Enterprise Architecture

It was recognized that improving schedule performance at the same time as technical architecture changes were implemented required new relationships among management and technical organizations. The cognizant program managers worked together to increase the speed of development by creating an iterative build-test-build process. Their goal was to more closely connect academic and government laboratory research and development with engineering integrators and operational users. This required changing the dependencies and responsibilities for technical tasks and re-allocating decision authorities among participating organizations. Changes to the development process were implemented in parallel with changes in the organizational structure. In an evolutionary (over several years) pattern, PEO-level management evaluated technical information dependencies and information requirements for the evolving system, crafting organizational dependencies and interfaces to address them.

Implementing the System Solution Within the Sociotechnical System

Program Office Architecture

Steps were taken to increase the richness of the dependencies between the Advanced Systems Technology Office (ASTO), responsible for developing advanced sonar technologies, and the submarine sonar systems program office (PMS4252). These changes were documented in the semi-annual PMS4252 Acoustic Program Plan (APP; Naval Sea Systems Command [PMS4252], 1994):

The original dependency:

> "ASTO has traditionally provided one of a kind systems … and then transitioned them to NAVSEA PMS5252 in the form of paper algorithm designs which, in turn, are provided to contractors for implementation."

The new dependency:

> "A common test bed and common, if not identical, deployable hardware will be created so that 6.3 developed capabilities can be easily transitioned to 6.4.
>
> PEO(USW) ASTO and PMS425 are coordinating the development of a concept that would enable expeditious fielding of an advance sonar processing concept into fleet systems."

A bulletized list of specific activities to implement this new coordination was also listed. Included in these architectural decisions was the goal of connecting the advanced development process to operational users (the fleet):

> "The approach of beta testing is fundamentally a means to allow the user (the fleet) to get a feel for a capability (new functions) and quickly provide feedback to the developers as to suitability before significant investment. It also provides a more expedient means to work out the operational concept in a forum well suited to define usability. Additionally, it may prove an effective means to introduce capability in parallel with new systems development, conducted by the commodity manager, PMS425, without long and costly changes to existing MILSPEC systems."

Evolution continued to an Integrated Project Team-Working Group structure which fundamentally changed organizational dependencies within and across industry, academia, and government.

Working Group Dependencies

The new development process was named "Advanced Processing Build" (APB). Early APBs were numbered sequentially (e.g., APB-1); later they were numbered according to the year in which they were developed (e.g., APB-98 for 1998). As indicated above, the ASTO-PMS4252 developed process evolved into an extensive set of interrelated working groups. The program managers actively identified and managed intra- and inter-working group dependencies and dependencies between the WG and participating organizations.

In an early example of this management, one ASTO program manager, in the face of confusion among the working groups, identified specific dependencies and the means by which they should be satisfied. These were laid out in a long directive to the WG, transmitted via email (Zarnich, 1996).

In this note, he specified three working groups, their work tasks, the technical and programmatic uncertainties surrounding them, and his decision rationale. He specified coordination dependencies with other WGs, the Navy Lab, a commercial contractor, and expected dependencies within the WG itself. He also directed the addition of WG members in order to ensure the right level of technical expertise was involved. Included in these directions were WG charters, performance milestones, work products, and deadlines. It is important to note that this directive memo included a traditional project plan in Microsoft Project format.

Resistance to New Dependencies: APB-T(01)

After the first two APB development cycles (APB-1[98] and APB-2[99]), the Program Executive Officer, Submarines (PEOSUBS) directed the extension of the new development process to the Combat Control System (CCS). The new process was named "APB(T)," and the initial development cycle was to be APB(T)-01. This new process was implemented by a different program office using the sonar APB model and involved a similar set of actions at the program manager level. New technical architecture choices were made, new dependencies identified and new organizational architecture (dependencies) implemented.

The new architecture created new dependencies, which caused friction and resistance. Specifically, the architecture changes shifted technical responsibilities and authorities. The Navy lab and its main contractor were particularly affected, and their response was to ignore the new dependencies, which resulted in poor performance of the program in shifting to the new APB model.

Specifically, assigned roles were disregarded, and work products were not delivered, which impacted the work products of other parts of the organization (Navy Program Manager, 2001). As an example, over a two month period, attempts were made to get software artifacts delivered to organizations responsible for integration, but the Navy

laboratory was late and, in one case, delivered obsolete software. As an example, one participant noted,

> The nature of the support needed is sufficiently broad, and dynamic, that a cooperative interactive engagement process is more appropriate than simply throwing request lists, and return questions about the items, over the transom to one another. While it's true that the items need to be clearly documented, even in an interactive process, unfortunately it's also true that the process can be ground to a crawl easily—and seemingly very technically and rigorously proper, if the participants don't all want to drive toward timely success. Establishing that motivation for mutual success among the engineering team typically isn't all that difficult—most of them are stimulated simply by the technical issues at hand and their natural desire to solve them. The leadership challenge in those circumstances is usually no more complex than making clear that timely success is the desired outcome and encouraging the cooperative engagement. Hopefully the leaders of the engineering team members are encouraging, rather than discouraging, that positive type of interaction.
>
> … executive-level folks all playing project facilitator, … clearly isn't reasonable (although occasionally it still may feel great). We'll continue to provide encouragement to our troops to strive for the cooperative interaction, hopefully yours will be provided similar encouragement from their bosses. (Navy Contractor, 2001)

Summary

Initial observations on these three simple examples highlight the focus of these Program Managers on dependencies and interactions. Based on interview data and email data records, they were mainly focused on implementing processes that increased outside, or non-traditional, participants in the development process. The expectation was that the new participants and a different process architecture and organization architecture would bring more objective evaluation of technical alternatives and, therefore, result in improved system performance.

However, their focus was on outcomes and on getting the “right” participants connected to each other. It was a very evolutionary process, where membership was increased or decreased based on immediate need, where WGs were established and disestablished as need dictated. The initial examination of this program’s history highlights the underlying importance of dependencies and attention to them.

Framework for Design & Control of Coordination

The discussion to this point has highlighted the importance of dependencies and interactions in programs, particularly in a dynamic or complex environment. It further argued that existing measures and control mechanisms do not fundamentally address dependencies and interactions, and therefore there is a significant opportunity to improve upon existing measures and control mechanisms. While this work is preliminary and ongoing, these points will be addressed in the following sections based on work recently completed or underway.

Characterizing and Measuring Dependence

A measurement and control system based on dependence and interaction must start with a characterization of dependence. Starke (2015) identified a set of eight characteristics of activity dependence and 21 corresponding measures derived from a review of the

literature and expert discussions (see Table 1). No measure was defined for the characteristic awareness, since awareness of a dependence itself is a precondition to be able to assess the dependence. Starke further attempted to validate the set of characteristics and their respective measures through a survey with 138 participants in a workshop. The work was considered preliminary, but he did establish that all the characteristics and measures were reliable with the exception of *Closeness and Degree of Mutuality*.[1] This characterization of dependence is a first step in the development of a comprehensive system of measurement for programs.

A key objective of a dependence measurement system must be to identify indicators of program behavior that can be linked to superior program outcomes. In the end, if it is to be useful (and used), it must demonstrate its ability to predict future program outcomes more accurately than existing measures. Consequently, it must comply with a number of requirements:

- It must be able to be instrumented so as to be practically and sustainably implemented in a performance measurement system.
- It must have a clear sampling approach, frequency, unit of analysis, etc. in order to produce reliable results.
- It must have a clearly-defined measurement process and ideally be indexed to current measurement and control systems in order to assess its predictive power relative to existing approaches.

This ongoing work is part of a larger effort that aims to revise and further develop these measures and develop an instrumentation method for gathering empirical data on dependence in programs and its impact on program control and performance. This work supports a larger agenda of determining whether it is possible to improve program assessment and control methods and tools beyond those currently in use.

[1] These two characteristics and their corresponding measures in particular have strong pooled dependence traits, and were considered to be not well-suited to the experimental methods used in the validation.

Table 1. Measures of Dependence in Programs

(Starke, 2015)

Dependence Characteristic	Description	Illustrative Measures
Awareness	The extent to which the interdependence is recognized within the process.	*No measures defined*
Closeness	The extent to which the actions of dependent activities have an immediate effect on each other.	Component connection within the product Flexibility of budget Number of activities altering data
Degree of mutuality	The extent to which the dependent activities have equal need for each other.	Difference in amount of needed information Difference in activity priorities Difference in data usage
Feedback mechanism	The way feedback is passed between dependent activities.	Frequency of scheduled information exchanges Frequency of scheduled budget reviews Number of times data is needed
Impact	The extent to which not fulfilling the dependence in the desired manner affects the dependent activities.	Fraction of activity dependent on input Rework caused by faulty input Excess capacity Specification connectedness
Satisfaction criteria	The criteria necessary to fulfill the dependence.	Understanding of what is necessary to fulfill the dependence Consensus on what is necessary to fulfill the dependence
Strength	The amount of required interaction as a direct result of the dependence.	Volume of necessary exchanged information Variability of costs Number of shared components Degree of concurrency
Urgency	The time-criticality for fulfilling the dependence.	Milestone flexibility Variance of activity duration

Coordination in Response to Dependence and Architecture

Having characterized and developed measures of dependence, the next step is to demonstrate how dependence plays a role in program coordination and control processes. Based on Moser, Grossmann, and Starke (2015) and Starke (2015), a framework was developed to demonstrate the mechanisms whereby dependence is satisfied in programs (see Figure 1) Dependence is driven by two sources of need: *Flow* and *Pool* causes. A flow cause of dependence results from the need for results or information from another task. A pool cause of dependence results from the need for a resource shared by another task. They both result in a demand for interaction.

Awareness of the dependence and allocation of attention are the major factors influencing how or if any interaction takes place. The volume, timeliness, cost, and quality of the interaction all have consequences regarding the satisfaction of the dependence. Dependency management, or coordination, may influence the demand itself, the awareness and the allocation of attention, as well as the interaction. Classic dependency management techniques seek to improve the awareness of the dependence (e.g., CPM or DSM) or improve the interaction (e.g., action plans or standardization).

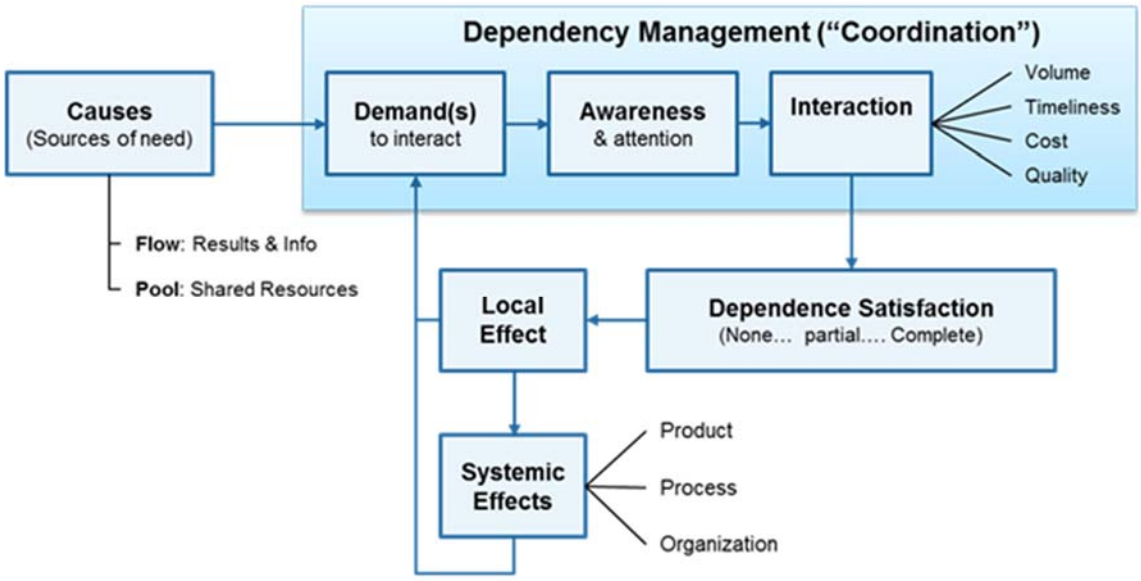

Figure 1. Mechanisms of Dependence From Cause to System Effects

Whether the dependence is satisfied will determine the local effects. This in turn influences the systemic effects. Local effects are the immediate consequences for the tasks (e.g., delay, costs, and rework) and for the individuals (e.g., frustration or establishment of trust). Systemic effects influence the significance of the local effect on product quality, the process as a whole, and the organization. These effects in turn can lead to a change in the remaining demand to interact. If the dependence is fully satisfied, the demand is effectively eliminated, and thus no demand to interact remains. If the dependence is only partly satisfied or not at all satisfied through insufficient interaction, demand to interact may decrease or even increase.

The Health of Interactions as an Early Indicator of Overall Performance

The framework shown in Figure 1 is a simple closed-loop feedback control system. As such, it lends itself to the development of a control system for program management based primarily on dependence. This alone represents a significant departure from existing tools and methods for program measurement, management and control. Current program planning and control practices (if sustainable) are necessary for governance, but may simultaneously act as a straightjacket on learning, depending upon the judgement of program and team leaders to make strategic adjustments. Perhaps an approach that relies on the basic characteristics of dependencies within a program could provide sufficient insights to free up critical program control capacity to enable more effective handling of exceptions, learning, and improvisation within the program.

The emergent and actual performance, in contrast to detailed baseline plans, reflects the gap between detailed control of tasks and strategic management of organization and interactions. Human teams take time and experience, and often fail to learn new habits of interaction. Simply completing one's own work according to finely separated work packages is not sufficient for system performance.

If a dependence-based control system is to make a difference, the treatment (coordination) of the interaction should be driven by the nature of the dependencies amongst tasks, where they fall across the organization, and the pattern of demands they place on teams. Teams would consequently be challenged to adjust their interactions so as

to be aware, pay attention, select amongst demands within limited capacity, and to perform. Important questions for teams, implied by the dependence-based framework include the following:

- Have the teams prioritized and paid attention to quality?
- Are defects/issues even noticed?
- If so, how does each team respond and make a decision on how to proceed?

Whether this framework and measurement approach is successful in spurring this kind of activity is the focus of ongoing research, and cannot be conclusively reported at this point in the process. Nevertheless, there is optimism based on not only literature reviews, but also anecdotal empirical evidence. The aim of this study is to collect systematic empirical evidence to assess the validity of the dependence-based approach to program activity measurement and control.

Conclusion, Limits, and Future Research

This paper has demonstrated a set of measures and an emerging measurement process for characterizing the fundamental elements of complex commercial, civil, and defense programs and projects. It focused specifically on the interactions and interdependencies that exist between the product system and social system. It identified implications for the execution of programs and future research relating to program management based on insights gained from this measurement approach.

The validation of this research on engineering projects as sociotechnical systems will require the instrumentation of performance during complex program planning and execution. The intent is to use this paper's representation of dependence to observe projects in progress to test the dependency model's practicality and usefulness. The paper doesn't present an analysis and conclusions because the work is underway.

Future work will require the preparation of a system to measure the demands on and the attention of teams across product, process, and project organization. The responses to dependence will be correlated to local and systemic performance. Additionally, experiments to test the effect of increased awareness of concurrent and mutual dependence on local and systemic performance of the engineering project will be needed. Generating sufficient data to validate this approach from multiple programs will be a lengthy process, but sample identification is already underway. Early experiments using the measures and framework discussed in this paper are promising, but more systematic data will ultimately tell whether this approach addresses the shortfalls in existing methods that have been identified.

References

Allworth, J. (n.d.). The 787's problems run deeper than outsourcing. Retrieved from https://hbr.org/2013/01/the-787s-problems-run-deeper-t

Board, I. S. (2000). *IEEE recommended practice for architectural description of software-intensive systems* (IEEE Std 1471-2000). New York, NY: The Institute of Electrical and Electronics Engineers.

Boeing 787 Dreamliner. (2016, April 4). In *Wikipedia*. Retrieved from https://en.wikipedia.org/wiki/Boeing_787_Dreamliner

Cantarelli, C. C., Flyvbjerg, B., Molin, E. J., & van Wee, B. (2010). Cost overruns in large-scale transportation infrastructure projects: Explanations and their theoretical embeddedness. *European Journal of Transport and Infrastructure Research, 10*(1), 5–18.

Christensen, D. S. (1994). Using performance indices to evaluate the estimate at completion. *The Journal of Cost Analysis, 11*(1), 17–23.

Clark, D., Sollins, J., Wroclawski, J., Dina, K., Kulik, J., Yang, X., … *Chiappa, N. (2004). New arch: Future generation internet architecture.* MIT and USC.

Clark, D., Wroclawski, J., Sollins, K., & Braden, R. (2005). Tussle in cyberspace: Defining tomorrow's Internet. *IEEE/ACM Transactions on Networking, 13*(3), 462–475.

Denning, S. (n.d.). What went wrong at Boeing? Retrieved from http://www.forbes.com/sites/stevedenning/2013/01/21/what-went-wrong-at-boeing

Dillard, J. T. (2003). *Centralized control of defense acquisition programs: A comparative review of the framework from 1987–2003*. Monterey, CA: Naval Postgraduate School.

Fleming, Q. W., & Koppelman, J. M. (1996*). Earned value project management* (1st ed.). Project Management Institute.

Fleming, Q., & Koppelman, J. (1994). The essence and evolution of earned value. Transactions of AACE International, 73.

Gantt, H. L. (1903). A graphical daily balance in manufacture. *Transactions of the ASME, 24*, 1322–1336.

GAO. (2006). *Defense acquisitions: Major weapon systems continue to experience cost and schedule problems under DoD's revised policy* (GAO-06-368). Washington, DC: Author.

GAO. (2009). *Assessments of selected weapon programs.* Report to congressional committees (GAO-09-326SP). Washington, DC: Author.

Garlan, D., & Shaw, M. (1993). An introduction to software architecture. In V. Ambriola, & G. Tortora (Eds.), *Advances in software engineering and knowledge engineering, volume 1.* New Jersey: World Scientific.

Gates, D. (2016, January 5). Will 787 program ever show an overall profit? Analysts grow more skeptical. Retrieved from http://www.seattletimes.com/business/boeing-aerospace/will-787-program-ever-show-an-overall-profit-analysts-grow-more-skeptical/

Goldratt, E. M. (1997). *Critical chain [A business novel].* Great Barrington, MA: North River Press.

Henderson, R. M., & Clark, K. (1990). Architectural innovation: The reconfiguration of existing product technologies and the failure of established firms [Special issue: Technology, Organizations, and Innovation]. *Administrative Science Quarterly, 35*(1).

Kelley, J. E., Jr., & Walker, M. R. (1959). *Critical-path planning and scheduling*. Paper presented at the Eastern Joint IRE-AIEE-ACM Computer Conference (pp. 160–173). ACM.

Khamooshi, H., & Golafshani, H. (2014). EDM: Earned duration management, a new approach to schedule performance management and measurement. *International Journal of Project Management, 32*(6), 1019–1041.

Lipke, W. (2003). Schedule is different. *PMI CPM Journal*, *The Measurable News*.

Lipke, W. (2004). Connecting earned value to the schedule. *The Measurable News, 1,* 6–16.

Lipke, W. (2012). Speculations on project duration forecasting. *The Measurable News, 3,* 3–7.

Moser, B., Grossmann, W., & Starke, P. (2015). Mechanisms of dependence in engineering projects as sociotechnical systems. In Transdisciplinary Lifecycle Analysis of Systems: *Proceedings of the 22nd ISPE International Conference on Concurrent Engineering* (Vol. 2, p. 142). IOS Press.

Moses, J. (2006). *Overview of organizational structures and system architectures* [ESD.342 class lecture notes]. Cambridge, MA: MIT.

Naval Sea Systems Command (PMS4252). (1994, September 30). *Acoustic program plan*.

Navy Contractor. (2001, March 15). More NUWC support items.

Navy Program Manager. (2001, March 5). NUWC impact to APB(T)-01.

Patanakul, P., Kwak, Y. H., Zwikael, O., & Liu, M. (2016). What impacts the performance of large-scale government projects? *International Journal of Project Management, 34*(3), 452–466.

Starke, P. (2015, June). *A New approach to understanding and measuring task interdependence* (Master's thesis). Munich, Bavaria, Germany: Technische Universität München.

Ulrich, K. T., & Eppinger, S. D. (1995). *Product design and development*. New York, NY: McGraw-Hill.

Wilson, J. M. (2003). Gantt charts: A centenary appreciation. *European Journal of Operational Research, 149*(2), 430–437.

Zarnich, R. W. (1996, September 3). Novel1.doc.

ACQUISITION RESEARCH PROGRAM
GRADUATE SCHOOL OF BUSINESS & PUBLIC POLICY
NAVAL POSTGRADUATE SCHOOL
555 DYER ROAD, INGERSOLL HALL
MONTEREY, CA 93943

www.acquisitionresearch.net