Systems Engineering: Have we lost our Competitive Edge?
A Consideration of the Dynamics of Systems Engineering Projects

K. Triantis (triantis@vt.edu), H. Rahmandad (hazhir@vt.edu), W. Vaneman (wvaneman@vt.edu)
Virginia Tech
Grado Department of Industrial and Systems Engineering
System Performance Laboratory
7054 Haycock Road
Falls Church, VA 22043-2311W. Vaneman (wvaneman@vt.edu)
Virginia Tech
Grado Department of Industrial and Systems Engineering
System Performance Laboratory
7054 Haycock Road
Falls Church, VA 22043-2311

Abstract

Recent media reports include several large systems engineering failures. These failures are especially alarming given that they span different sectors (i.e., shipbuilding and space systems), and are not isolated to one firm. Therefore we need to ask: Have we lost our systems engineering competitive edge? What can the systems engineering discipline do to correct the apparent discrepancies that appear to be at the root cause of these failures?

A systematic framework that represents current system engineering practices and integrates different factors that impact its performance into a unified view is not currently available. We introduce some of the key concepts of this integrative framework by borrowing from the management and system dynamics literature. This framework facilitates the modeling of the systems engineering process for the purpose of understanding, assessing, and potentially improving its performance. Our framework brings together the basic mechanics (e.g., task completion, testing, scheduling, and costing) and the human elements (e.g., skills, incentives, and employee turnover) inherent in system engineering projects. We highlight major feedback processes crossing multiple stages of the process and leading to cost and budget overruns. We demonstrate how this framework can organize and connect multiple sources of failure in the systems engineering process.

Key words and phrases: life-cycle systems engineering design, systems engineering projects, Intra phase, inter phase, multiple project dynamics.

1.0 Research Motivation and System Engineering Process Issues

From the beginning of man's history of developing large systems and architectures, until the 1950's, the responsibility of traditional systems engineering function such as architectural trades, cost estimation, program management, etc. fell upon the shoulders of the architect and the civil engineer. During this era, the World realized several great and timeless systems (e.g., the pyramids, the Roman Aqueducts, the Hoover Dam, the Brooklyn Bridge, and the Empire State Building to name a few). While these architectures were great engineering feats, they exhibit some degree of engineering complexity they only involved a limited number of engineering disciplines. During World War II, project managers and chief engineers provided the oversight to more complex systems that required a broader range of engineering disciplines. The Allies enjoyed additional engineering successes with large scale shipbuilding, aircraft, and weapon programs that it took win that war.

By the mid-1950's, systems were evolving with a greater degree of complexity. The advent of the computer ensured that the World would be able to perform missions and tasks that heretofore were unimaginable. A few examples include the development of: nuclear powered ships and submarines; missiles with nuclear weapons; and satellites and manned space vehicles. These new systems exhibited a different type of complexity, hence requiring a wider degree of involvement from the various engineering disciplines. The importance of these systems also required a high degree of confidence that they would perform as designed, be delivered within budget, and on time. The discipline of systems engineering evolved to develop the requisite design processes and act as the integrator between the various engineering disciplines. The successes enjoyed by the systems engineering discipline have primarily been single systems often with many sub-systems that were responsible for executing well defined missions and tasks.

Recently, the media has been reporting about several large systems engineering failures (Taubman 2007). These failures would be troublesome if they spanned one large system developed by a single firm, or one specific industry (e.g., the shipbuilding industry). However, these reports are especially alarming given that they span several large system sectors (i.e., shipbuilding and space systems to name a few), and are not isolated to one firm. These apparent breakdowns have motivated us to ask: What has happened to the systems engineering discipline? Have we lost our systems engineering competitive edge? And what can the systems engineering discipline do to correct the apparent discrepancies that appear to be at the root cause of these failures?

A few government and industry experts, from the shipbuilding and space industries, have recently published articles in various trade magazines, and papers, speculating why the systems engineering discipline is failing. A topical listing of their theses is portrayed in Table 1. The common themes from across the literature include: the need to return to systems engineering basics; poor cost estimation; problems with the industrial base; customer-developer relationships; erosion of domain knowledge; acquisition process and program management failures; and problems with program oversight. Other themes not common across all of the literature include: lack of risk management, unrealistic expectations associated with technological innovations; and funding instability.

Winter (2007)	Nowinski and Kohler (2007)	Smith (2007)
Systems Engineering: Lack of rigor; non-effective allocation of resources in the process.	Lack of systems engineering basics.	Lack of systems engineering basics.
Systems Engineering Management: Lack of focus on its core elements; not effective maintenance management.	Lack of program management basics.	Weakening of program management.
Erosion of Domain Knowledge: Lack of knowledge transfer; lack of employee loyalty; issues with reward and incentives structures.	Loss of domain expertise.	Loss of domain expertise.
Provider Customer Relations: Mutual lack of understanding of business processes and cultures; lack of customer loyalty; changing requirements definition; the impact of generated error rates.	Ineffective team building.	Ineffective Government/ industry teaming.
COTS Technologies: Unrealistic expectations; imposition of constraints (reduction of the design feasible space) without appropriate adjustment to requirements.	Lack of innovation	
Industrial Base Erosion: Underestimating economies of scale dynamics; workforce erosion; infrastructure depletion.	Erosion of industrial base and the motivation to maintain the base.	Erosion of industrial base.
Ineffective Cost Estimation: Underestimating costs;	Poor cost estimation.	Poor cost projection.
Design Constraints: Lack of definition of the design solution space.		
Acquisition Process Failures: Ineffective incentive structures; ineffective budgeting process.	Inability to lead acquisition process	Acquisition management and process failures
Requirements Definition: The impact of using iterative vs. waterfall approaches.	Tight, non-iterative system requirements.	
Lack of program oversight.	Lack of oversight	Lack of oversight
Adverse Impact of Competition.		Competition.
		Funding instability.
	Lack of commitment to the Mission	
	Ineffective Risk Management Procedures	

Table 1: Potential Sources of System Engineering Failures

Additionally, a systematic framework that represents current system engineering practices and integrates different factors that impact its performance into a unified view is not available. The purpose of this paper is to discuss some of the key elements and concepts of this integrative framework by borrowing from the management and system dynamics literature. This integrative framework is essential because: a) Tight connections exist between different stages of the systems engineering process, e.g. the contracting process impacts incentives for resource allocation across different life-cycle stages, which in turn impacts cost-schedule-functionality

tradeoffs. Therefore in managing the systems engineering process we are confronted with a complex nonlinear dynamic system in which fixing one part of the process may, or may not, lead to: a) overall performance improvement; or b) Effective improvement actions relying on comparing alternative interventions. Such comparisons require a systemic framework that can quantitatively capture the interconnected system engineering stages and the impact of alternative corrective actions.

We assume the systems engineering life-cycle process as defined by Blanchard and Fabrycky (2006). This is the traditional systems engineering process, encompassing conceptual design, preliminary design, detailed design and development, production/construction, utilization, evaluation, and support, and, finally, phase-out and disposal. This systems engineering life-cycle process definition is consistent with current systems engineering practices in government and industry. Furthermore, we will look at issues that manifest themselves within phases of this process (intra-phase), among phases (inter-phase), and with concurrent multiple projects (system of systems).

The framework will facilitate modeling of the systems engineering process for the purpose of understanding, assessing, and potentially improving its performance. Our framework brings together the basic mechanics (e.g., task completion, testing, scheduling, and costing) and the human elements (e.g. skills, incentives, turnover) in system engineering projects. We highlight major feedback processes that cut across multiple stages of the process and are at the center of significant cost and budget overruns (e.g. schedule pressure may trigger an increase in parallel work which leads to higher a error rate (given the interdependencies between tasks) and requires future rework and thus increases delays and pressure further). We demonstrate how this framework can organize and connect multiple sources of failure in the systems engineering process, and how it can be used to avoid those problems in the first place.

Furthermore, we can address other issues not presented in the literature. For example what role does system complexity have in these failures? Systems today are very complex, and are required to perform a larger array of missions and tasks. In this net-centric World, an exhaustive listing of missions and tasks, as well as systems that may have to be interfaced with during the life-cycle cannot be predicted during the systems design phase. Many of today's systems are being developed considering the capabilities that they add and not for a standardized set of missions and tasks.

The remainder of this paper is organized as follows: Section 2 provides a quick overview of recent studies that expand on some of the issues associated with the reasons why system engineering implementations can fail. Section 3 describes some constructs from the literature that may explain how some of the failures occur. These constructs would form the initial foundation for the integrated analytical framework. Section 4 concludes with recommendations for future research.

2.0 Background

We provide a brief overview of the recent literature that includes commentary on systems engineering process issues, issues on analytical methods used, the consideration of exogenous factors, the system of systems concept, and the interface between system dynamics with systems engineering. This overview is by no means an exhaustive description of what exists in the literature.

2.1 Systems Engineering Process Issues

Haskins (2007) finds limits to the application of systems engineering frameworks, noting that twenty years ago the pervasive thought was that growth in the practice of systems engineering would be focused on the increased number and diversity of problems to which it is applied. However, the current practice of systems engineering primarily relies on the creation of technological alternatives that are associated with engineered solutions. Within this narrow perspective, there are issues related to the application of structured analysis techniques, the appropriate representation of requirements, and the fact that a majority of systems are software intensive.

Eriksson, Borg, and Borstler (2008) emphasize that the implementation of traditional systems engineering processes suffer because of their use of classical structured analysis techniques that provide little support for achieving high levels of reuse. This in part due to the fact that the top down process of traditional functional decomposition does not have any inherent mechanisms for developing requirements that map effectively to existing reusable components.

Additionally, Bahill and Henderson (2005) discuss the importance of effective requirements definition by detailing problems in historically famous failures. For example, they blame the Tacoma Narrows Bridge collapse on the design engineers who reused the requirements for another existing bridge. However, they did not realize that this was the wrong bridge for that environment. Bahill and Henderson (2005) emphasize the importance of effective requirements development, verification and validation. If these activities are not performed adequately, this can cause failure in the system either individually, collectively, or in conjunction with other types of failures.

Kossiakoff and Sweet (2003) discuss the problems associated with designing software-intensive systems. For software-intensive systems where the software performs virtually all the functionality, such as in modern financial systems, airline reservation systems, and other information systems, they generally follow life cycles similar in form to the more traditional systems. However, the design of these systems involves a considerable amount of iteration and prototyping something that is not typically acknowledged up front.

One should also investigate issues that are above and beyond the technical problems associated with the implementation of the systems engineering process. For example, Haskins (2007) points to the need of focusing on the social aspects of the engineered solutions something that is not extensively practiced today. In part, this is what has led to the current emphasis on human systems integration (HSI) approaches. Haskins (2007) continues by arguing that a language for applying systems engineering to the social aspects of engineered solutions is not pervasive. This is partially due to the fact that system engineers lack training in the behavioral and social sciences as well as lack of on the job exposure to these disciplines.

One can use some of the recent systems thinking and modeling approaches to understand and improve the systems engineering process. For example, Bar-Yam (2003) has proposed that complex engineering projects should be managed as evolutionary processes that undergo continuous rapid improvement through iterative incremental changes performed in parallel. In this way one can focus on and link diverse small subsystems of various sizes and associations. In general, constraints and dependencies increase the system complexity and should be imposed only when necessary. In this evolutionary context, people and technology are agents that are involved in the design, implementation and function. Management's basic oversight responsibility is to create a context and design the process of innovation.

2.2 Issues with Analytical Methods Used

Recently, the literature has been critical of many of the traditional tools used in systems engineering. Issues with the use of analytical techniques such as IDEFϕ diagrams, quality function deployment (QFD), system dynamics modeling, highly optimized tolerance (HOT), COSYSMO, and for software intensive systems the user interface prototyping have been criticized and questioned for relevance.

Eriksson, Borg, and Borstler (2008) point out that the use of IDEFϕ diagrams are not understood by nontechnical stakeholders or even other engineering disciplines. They suggest that this constitutes a severe problem, since systems engineering is the means by which stakeholder needs and expectations are translated and communicated to other engineering disciplines. These authors recommend the use of case modeling that can be easily communicated to both nontechnical and other engineering stakeholders. In addition, the authors recommend having domain engineers (i.e. mechanical, electrical, software engineers), as opposed to systems engineers, take responsibility for specifying subsystem requirements. They believe this will encourage domain engineers to analyze the origin of subsystem requirements that will give them a total system understanding, while systems engineers can facilitate the process by maintaining the original intent of the system-level requirements. While Eriksson, Borg, and Borstler (2008) point out potential weaknesses with IDEF diagrams, we believe that the thesis of de-emphasizing the role of the systems engineer at the subsystem level is fundamentally flawed. It is the role of the systems engineer to optimize the performance of the entire system and not one sub-system. Allocating subsystem requirements to a specific domain will optimize the subsystem, and will most likely sub-optimize the system as a whole.

Hari, Kasser and Weiss (2007) examine another popular approach used in systems engineering - Quality Function Deployment (QFD). They state that QFD has a lot to offer with respect to translating the requirements of new products, but when this approach is used to specify the requirements of complex systems, it has been found to exhibit a number of deficiencies. The authors point out that the current development paradigm for complex systems is typically characterized by large cost overruns, schedule slippages, and performance deficiencies. The major contributor to these failures is inadequate requirements definition. When describing the requirements development process, they say that main problem arises when the needs are identified and converted to requirements. This is a simple statement yet hides a complicated requirements definition process that tends in reality to be iterative and often multi-phased and difficult to control Furthermore, performance requirements are many times expressed in a solution language not in a problem language, and tend to be incomplete, incorrect, and poorly written. These authors recommend that the current implementation of QFD be modified and evolved into a process for defining requirements for complex systems named Quality Requirements Definition (QRD).

Boppana et al. (2006) state that one could integrate three different analytical approaches: system dynamics (SD), highly optimized tolerance (HOT), and COSYSMO. This integration provides a promising future research direction that potentially could address the complexity in systems engineering process implementation. The reasoning behind this combination is to build on the strengths of all three approaches, i.e., SD for relatively detailed process modeling, HOT for coarse, higher-level modeling, and COSYSMO for calibration referencing. System dynamics (SD) can show emergent behaviors by accounting for interactions and feedback loops. Highly optimized tolerance (HOT) provides the cost, schedule, and performance within systems engineering with a relatively simple model. COSYSMO is a parametric cost model and uses data from past systems engineering efforts and subjective inputs from systems engineering experts. COSYSMO is calibrated with expert inputs and past experience and addresses cost, schedule, and performance, but cannot model emergent behaviors. The application of SD and HOT to an FAA Advanced Automation System program provided inconclusive evidence as to whether these

modeling approaches can provide useful insights into the complexity and emerging phenomena of the systems engineering process.

For software intensive systems, McConnell (1998) recommends that User Interface Prototypes are developed. These are defined as a mock-up of software that is created for the purpose of eliciting user feedback about the software's intended functionality including look and feel. This activity could be part of the requirements development process and would address Kossiakoff and Sweet's (2003) contention that it is the responsibility of systems engineering to analyze all requirements and specifications in detail first in order to understand them vis-à-vis the basic needs that the system is intended to satisfy. After this first step it is important to identify and correct any ambiguities or inconsistencies in the definition of the system capabilities. This type of prototyping could also solve some of the requirements stability challenges that have plagued information system designs.

2.3 The Consideration of Exogenous Factors

Briggs and Little (2008) find that when it comes to major decisions made in technical enterprises, understanding the environment of the enterprise, including its culture could significantly increase the likelihood of successfully implementing the systems engineering process. They state that the failure to consider the larger context in which technical decisions are made can enhance the risk of systems failures. They describe a situation where the decision about information requirements never took into account the people who used the information to make decisions. The authors argue that decision making processes should explore ways to explicitly consider a wide range of stakeholders that are broadly defined, and the social and cultural context within which these stakeholders operate. They focus what are usually considered to be exogenous factors that are some combination of temporal, financial, and cultural factors. These include: business time-to-market pressures to produce products in shorter time; economic pressures to lower the cost of products; and the wholesale merging of enterprises.

Lewis (2007) also emphasizes the importance of social aspects of the enterprise. However, he focuses on the importance of team behavior. He states that it is hard to predict group behavior by observing individual behavior and that one needs to be careful making projections about team behavior by focusing only at individual team member qualities.

2.4 System of System Considerations

Wojcik and K.C. Hoffman (2006) address the evolutionary and emergent behavior of Systems of Systems Engineering (SoSE). Enterprises that are involved with SoSE are typically complex, multi-agent enterprises or sets of enterprises exhibiting the characteristics of complex adaptive Systems. SoSE is typically only one aspect of an enterprise's activities, and the whole set of activities is mainly oriented towards accomplishing and supporting the enterprise's operational mission. This work proposes a unifying framework for understanding and modeling the organizational, technical, and system complexities across a range of enterprise types as major acquisition program initiatives are undertaken to provide improved operational capabilities.

Sage and Biemer (2007) state within government and industry, many systems are currently engineered, not as stand-alone systems, but as part of an integrated system of systems. According to the authors, a significant level of effort has been devoted in the past several years to the development, refinement, and acceptance of processes for engineering systems. Today, there are four standard processes that exist within present and past standards, i.e., EIA-632, IEEE 1220, ISO 15288, and MIL-STD-499C. These standard processes are used for the design of system of systems.

2.5 Interfaces of Systems Engineering with System Dynamics Literature

McLucas and Ryan (2005) argue that a detailed appreciation of how systems engineers define, analyze, specify, manufacture, operate and support complex systems could inform the evolution of system dynamics even though there are significant differences between the two disciplines. Their preferred approach integrates systems thinking, system dynamics modeling and systems engineering. This integrated approach could enable group model building and the formulation of effective models through top-down design and careful management of the complexity introduced at each stage of the model-building process. The integrated approach promises to invoke greater confidence in system representations that actually work.

3.0 Dynamics of the Systems Engineering Process

In this section we draw on extant literature to illustrate how dynamics of systems engineering process can lead to heterogeneous performance in systems engineering projects. In short, why many projects get into trouble where some others have succeeded? We focus on heterogeneity in performance because that is the real phenomenon of interest. If all projects did equally well, or poorly, this would have signaled little room for improvement. In fact, we are motivated to understand the dynamics of systems engineering process because there is hope in helping all system engineering projects achieve what is demonstrated as achievable by the most successful ones.

As indicated in the previous Section, systems engineering projects unfold over time as a result of interactions between multitude of technical, organizational, and political factors. Therefore the principles and tools developed for analyzing complex dynamic systems are directly applicable to the question at hand. We draw on those principles as developed in the fields of system dynamics and complexity to build a robust theoretical foundation for our research.

A critical insight from studying dynamic systems is that reinforcing feedback processes lie at the heart of growth and decline dynamics. These processes therefore can derive a wedge between successful and unsuccessful enterprises and lead to heterogeneity in performance. For example superior sales performance enables a firm to invest more in product development and marketing, and thus build better products and sell even more. Similarly, larger market share increases bargaining power and derives down costs, thus enabling a firm to gain further market share through lower price. In the next section we illustrate this principle with a stylized simulation model of an enterprise. We then draw on this insight to discuss the important reinforcing feedback processes that can impact systems engineering projects and lead to the observed heterogeneity in their performance.

3.1 Reinforcing Feedback and Performance Heterogeneity: An Illustration

Consider a simple system's engineering unit in charge of engineering tasks associated with the system's life-cycle development, e.g., requirements definition, prototyping, etc. New tasks flow into the stock of "*Tasks to Complete*" when assigned by the higher management through the scheduling of activities. These tasks flow out of this stock through "Task Completion" (See Figure 1; Variable names are italicized). For the moment, let us assume that *Task Completion* is a function of the number of people in the enterprise ("*Resources*"), their productivity, and the quality of their work, i.e. "*Fraction Acceptable*". Following previous research (Cooper 1994; Graham 2000; Ford and Sterman 2003; Nepal, Park et al. 2006; Taylor and Ford 2006) we note that productivity and quality are not constant, but depend on the schedule pressure. Schedule pressure compares the expected time to finish the current *Tasks to Complete,* given available resources and productivity, with the *Desired Completion Time*.

Schedule Pressure often increases (gross) productivity, as people work harder, spend more time at work, or cut down on training, and other perceived non-urgent activities. The productivity-

enhancing effect of schedule pressure is bounded and saturates at some level. On the other hand, if schedule pressure is very low, people produce less because there are few tasks to complete. These relationships are graphically illustrated inside Figure 1. The impact of schedule pressure on productivity creates a balancing feedback process: if *Tasks to Complete* grow, schedule pressure increases, that leads to higher "*Effect of Pressure on Productivity"*, which (everything else being equal) increases *"Task Completion"* and therefore reduces *Tasks to Complete* and balances the pressure. This balancing loop is specified as "*B: Work Harder"* in the Figure 1[1].

On the other hand, as one may expect, there are unintended side effects to *Schedule Pressure* as well. Under pressure people are more likely to take shortcuts and cut corners, leading to increased error rates in their work. Moreover, sustained schedule pressure leads to burnout and further decline in quality. Overall, these processes lead to a negative relationship between *Schedule Pressure* and the fraction of completed tasks that are acceptable (*Fraction Acceptable).* This relationship (illustrated in Figure 1) creates a reinforcing feedback loop: An increase in *Schedule Pressure* leads to lower *Fraction Acceptable*, and therefore reduces *Task Completion.* As a result the *Task to Complete* grows beyond what it would have been otherwise, and fuels even more *Schedule Pressure* (The reinforcing loop "*R: Rework"* in Figure 1). In fact, one may combine the impact of schedule pressure on productivity and quality, to calculate the net relationship between *Schedule Pressure* and *Task Completion.* This relationship suggests that up to some point schedule pressure is functional (leads to higher performance (i.e. *Task Completion*), as the balancing loop dominates) but after that it hurts enterprise performance as the reinforcing loop take over (See Figure 1a).

Now consider the behavior of two identical systems engineering enterprises, modeled as above. These enterprises are exposed to random streams of *Task Assignment* which have identical distribution, but two different realizations. One may expect that given their identical structures and same demand distribution (task assignment), the two enterprises should behave very similarly. However, this is not necessarily the case. Figure 2 shows one such experiment. We have graphed *Task Assignment*, *Task Completion*, and *Tasks to Complete* for each enterprise. While the first enterprise is successfully managing to complete all the tasks assigned in a timely fashion and thus keeps the stock of *Tasks to Complete* under control (Figure 2-a), the second enterprise collapses in this experiment (Figure 2-b).

The first enterprise remains in a relatively stable domain because, by luck, the stock of *Tasks to Complete* and therefore *Schedule Pressure* do not grow into the domain where the negative effects of the *Rework* reinforcing loop dominate. As long as the *Work Harder* loop is stronger, temporary random increases in demand are compensated by increased task completion, and the enterprise manages its workload successfully. However, the second enterprise, by chance, experiences a little longer exposure to high *Task Assignment* levels. At about time 40 the accumulation of *Tasks to Complete* starts to push the *Schedule Pressure* beyond functional levels. Once the reinforcing loop of *Rework* gains the upper hand the enterprise completes fewer tasks due to errors and rework, falls further behind the unrelenting stream of *Task Assignment* and faces even higher levels of pressure. The cycle continues, eroding the capability of the enterprise to produce as before (see the permanent decline in *Task Completion* in Figure 2-b), and leading to accumulation of unfinished tasks, late projects, and unsatisfied customers.

The divergence in the behavior of the two enterprises is not an anomaly. Figure 3 shows the *Tasks to Complete* for one hundred identical enterprises with same demand distribution, which only differ in random realization of *Task Assignment* values they face. Here a few firms avoid the collapse under the reinforcing feedback while others succumb at different times. These

[1] Loop names are used to summarize the basic idea behind a feedback process and are used as memory aids.

potentially harmful dynamics could be mitigated by different interventions, each essentially existing of removing a reinforcing loop or creating a control mechanism (balancing loop) to keep the reinforcing process in check. For example, a balancing mechanism can be added that endogenously changes available *Resources* (or *Task Assignment*) based on the current schedule pressure. Alternatively, by using more effective work processes (such as, iterative prototyping, effective translation of user needs to requirements, etc.), the enterprise members could significantly weaken the *Rework* reinforcing loop (potentially with the cost of also weakening the *Work Harder* loop).

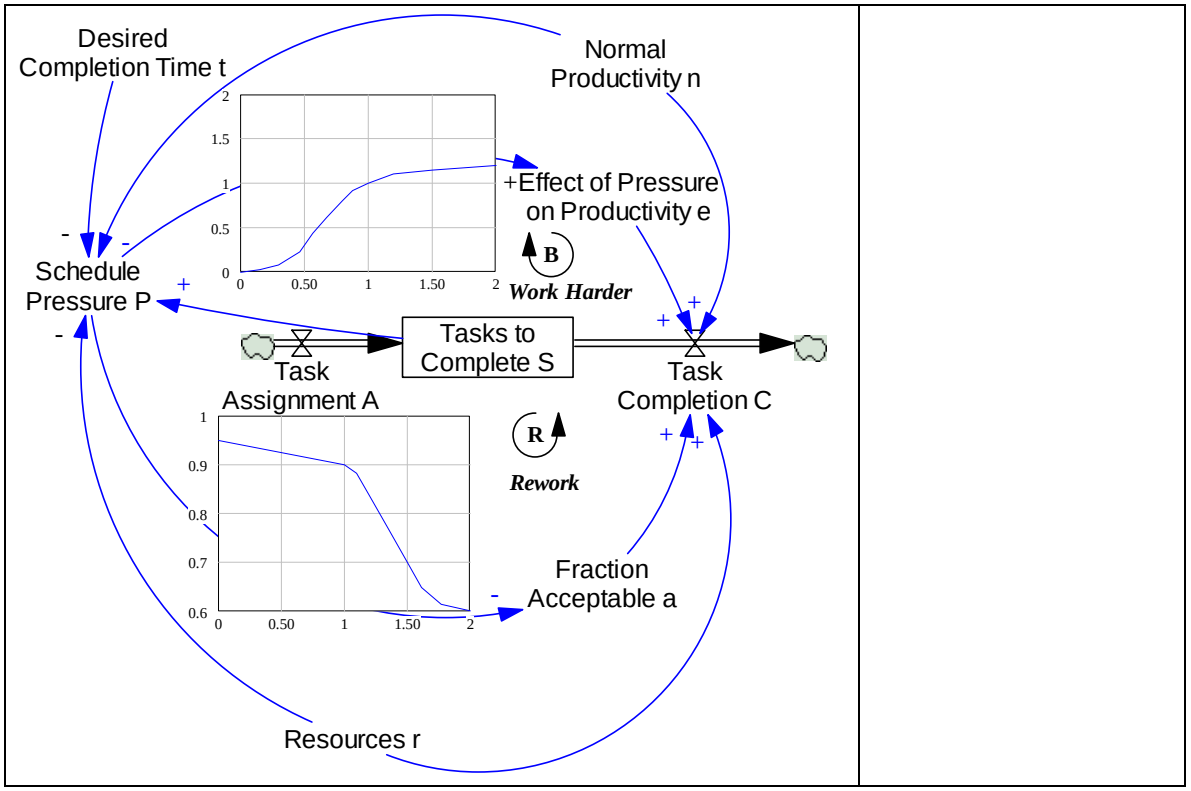

Figure 1: System Engineering Task Assignment and Completion

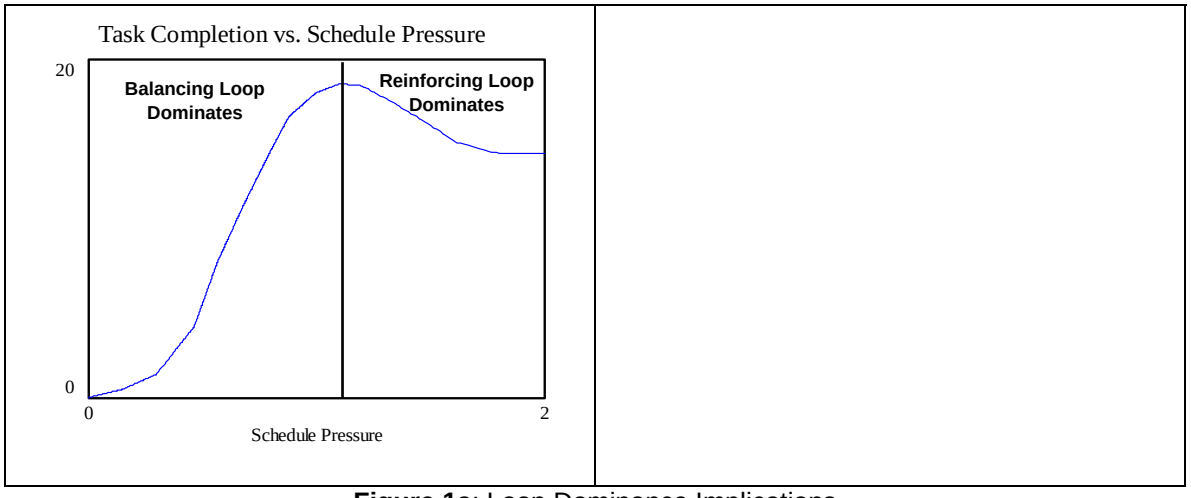

Figure 1a: Loop Dominance Implications

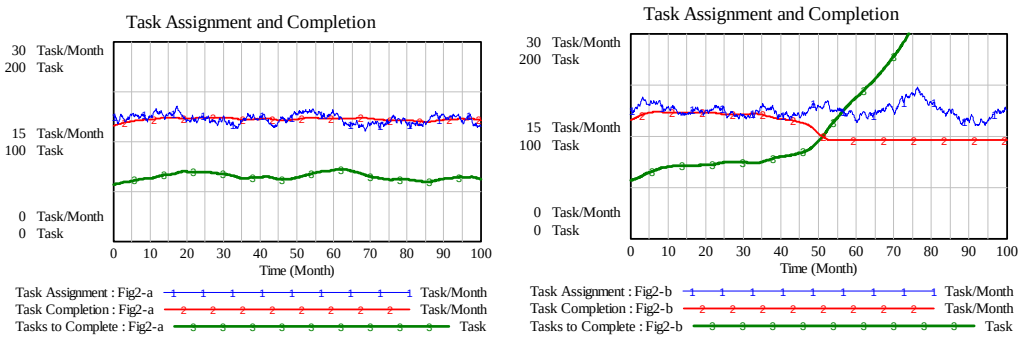

Figure 2: Performance of Two Systems Engineering Enterprises (a-Tasks to Complete Under Control; b-Tasks to Complete Growing)

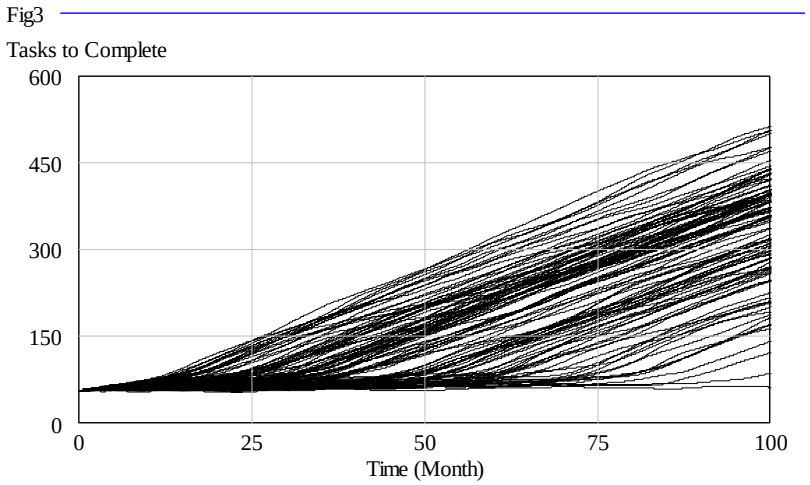

Figure 3: Tasks to Complete for Multiple Enterprises

This model is very simple and only intended to illustrate how reinforcing processes can lead to heterogeneity in the performance of systems engineering enterprises. The basic insight is that small random differences among otherwise similar enterprises (or projects, teams etc) can be magnified through reinforcing feedback processes. In other words, similar enterprises can perform very differently because of the internal dynamics they are embedded in, and not necessarily because of the exogenous factors, e.g. congressional budgeting process, system engineering labor market, etc., that impact their performance. This is potentially good news for those interested in improving the performance of the systems engineering process: if exogenous factors were the only driver of project performance, there would have been little leverage available to those in charge of the project. They would be at the mercy of the legislator, funding cycles, the economy, or other exogenous factors. However, once the reinforcing feedback processes that differentiate success and failure in systems engineering projects are identified, the managers are enabled to design mechanisms to control the undesirable effects of these processes or strengthen the beneficial feedback loops. In the next section we build on this basic insight and identify several reinforcing feedback processes that are in effect in the systems engineering process, and can potentially take over the dynamics of success and failure in these settings.

3.2. Feedback Processes across the Systems Engineering Process Lifecycle

In this section we draw on literature in system dynamics, project management, systems engineering, as well as our previous research in systems engineering and product development dynamics, to provide an overview of reinforcing feedback loops that can impact systems engineering projects' performance. This survey aims at a comprehensive survey of potential feedback processes. Therefore, most enterprises include the structure that leads to only a subset of these possibilities, and yet fewer of the feedback effects are strongly felt in any given enterprise. Detailed research at project or enterprise level is needed to identify specific processes most relevant to each enterprise.

These feedback processes cut across multiple levels of analysis: some are limited to a specific phase of the systems engineering process, some include interactions across multiple phases of the systems engineering process (e.g., the impact of the requirements definition in the conceptual phase is pervasive throughout the life-cycle of the system design), and some relate to interactions across multiple projects and enterprise capabilities. The distinction between these levels of analysis is useful because it enables different individuals at different positions to have an impact on project and enterprise performance. For example the reinforcing loops at the level of single phase are often endogenous to individual engineer, that is, a typical engineer has leverage points at her disposal that can change the strength and impact of those loops. The same individual may find few options to change the impact of loops that cut across multiple projects, and are only subject to interventions by enterprise leaders.

3.2.1 Intra-phase Dynamics

While they differ in technical content and complexity, all systems engineering design projects go through similar phases throughout their life-cycle. These include: conceptual design, preliminary design, detailed design, development, production/construction, utilization, evaluation, and support, and, finally, phase-out and disposal. Within these phases, a set of tasks are executed such as identification of customer needs, translation of needs into requirements, trade studies, testing, prototyping, measuring, training, maintenance, etc. Each phase of the systems engineering process includes multiple tasks often assigned to a team of individuals within the enterprise. The work within these phases usually follows a project structure: a relatively fixed set of tasks are assigned to a specific group to be completed within a planned schedule and cost. Therefore, regardless of the nature and complexity of the tasks, these phases embed several feedback processes that are documented in this discussion. Figure 4 provides an overview of these dynamics.

Task assignment and completion are the main ways through which the stock of tasks to do is changed. Different estimation processes are used to decide how long the completion of the current tasks take based on the available *Human Resources* and their *Productivity* and quality (*Error Rate*). The resulting *Scheduled Finish Time* compared to the current *Tasks to Complete* and available resources can lead to *Schedule Pressure.* Early studies showed that inclusion of the expected error rate in the project estimation process is critical but not always done rigorously. The error rate is critical because it wastes a fraction of tasks completed and requires additional rework that may not be discovered for a while, catching the project team by surprise later (Cooper 1980; Abdelhamid and Madnick 1983). For simplicity we have not explicitly shown the cycle of defect generation, discovery, and rework (rework cycle), yet we model its main ramifications through the link from the *Error Rate* to *Task Completion*. As we discussed in Section 3.1, schedule pressure impacts both productivity and quality of work, completing a balancing (*B1: Work Harder*) and a reinforcing (*R1: Rework*) loop. While moderate levels of schedule pressure increase output, high pressure can be dysfunctional, reducing the overall output given the disruptive impacts of lower quality and rework. One possible way to control excess schedule pressure is to bring new people to work on the current phase of the systems engineering project (*B2: Bring New Resources*). However, this controlling loop can have its own side effects as new

people are not up-to-speed with current project and thus have low productivity/quality, and their training takes away productive time from the experienced personnel (*R2: Skill Dilution*). Further negative impacts are observed if new hires disrupt the cohesiveness of the team and reduce productivity through additional communication costs. The resulting reinforcing processes are shown to be potentially harmful traps in many domains, including software development, where it has evolved into the Brook's law: adding people to a late project will delay it further (Brooks 1995). Schedule pressure can also disrupt a project indirectly when it contributes to burnout that leads to the attrition of key personnel and further pressure (*R3: Burnout*) (Cooper 1994).

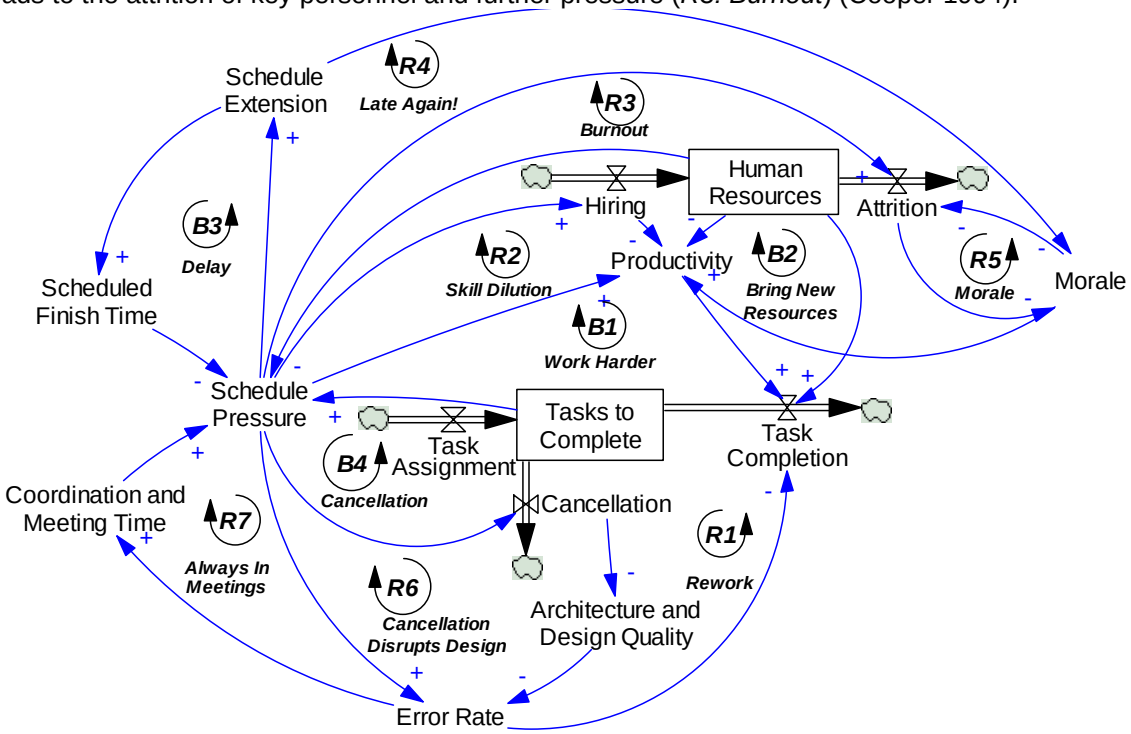

Figure 4: Intra-phase Systems Engineering Process Dynamics

Another control mechanism to relieve the schedule pressure is to extend the schedule for the current phase (*B3: Delay*). Besides the resistance to this solution that comes from the people involved in the next phases of the project or the client, delays may also have other subtle impacts. For example, in a recent study of a large IT product development organization, we heard about the negative impact of multiple delays on the team's morale (Rahmandad 2005). Given the impact of employees' morale on their productivity (Weakliem and Frenkel 2006) this can lead to another reinforcing loop, *R4: Late Again!.* Morale also has a reciprocal relationship with attrition: on the one hand lower morale leads to higher *attrition* probability (Johnsrud, Heck, and Rover. 2000), on the other hand, people whose friends have left are more likely to face a decline in their morale, completing another reinforcing loop (*R5: Morale*).

Reducing the *Tasks to Complete* through cancellation of parts of tasks for a phase is another way to reduce the schedule pressure and get a troubled project back on track (*B4: Cancellation*). On the other hand, the interdependencies between different tasks mean the quality of doing one depends on the others as well (Sobrero and Roberts 2001). Therefore late stage cancellation of some of the tasks may negatively impact the quality of the other pieces of work and increase schedule pressure inadvertently (*R6: Cancellation Disrupts Design*). Finally, low discovery rework due to high error rates often triggers additional administrative and coordination work including new meetings, re-estimations, and scores of e-mails and phone calls. These all take away from productive work otherwise spent on task completion, and thus increases schedule pressure (*R7: Always in Meetings)*.

In summary, there are several reinforcing processes within each phase of the systems engineering process. These processes, if active in an enterprise, can lead to increasing delays, cost and schedule overruns, and quality issues. Once these loops are triggered, they can become self-sustaining and continue to erode the performance of the phase involved. As a result, they can make the difference between successful and unsuccessful projects and lead to diverse performance outcomes from otherwise similar projects. Lyneis and Ford (2007) provide a more detailed treatment of many of these dynamics.

These dynamics are largely dependent on schedule pressure inside the group and highlight an important tradeoff: on the on hand low levels of schedule pressure point to low resource utilization. On the other hand, very high-levels of schedule pressure are disruptive through many second order effects. Once the enterprise moves to this dysfunctional region of schedule pressure (i.e. right hand side of the peak in Figure 1a), it is very hard to stop further deterioration of the performance because of the reinforcing nature of the feedbacks involved. This leads to a critical insight: most systems engineering projects face a tradeoff between resiliency and efficiency. Maximum output is achieved when schedule pressure is at the peak of the schedule pressure-task completion curve. However, keeping the schedule pressure at that peak level is very risky as any unexpected disruption can tip the enterprise into a state of making more errors, losing skills and morale, and thus achieving less despite working harder. Such tipping dynamics are documented in different domains from air traffic control (Rudolph and Repenning 2002) to software development (Rahmandad and Weiss 2008). It is required to sacrifice some efficiency by lowering schedule pressure below peak levels to increase enterprise resiliency against random events and unplanned changes. Larger resiliency margins are needed for projects with higher probability of unexpected disruptions.

3.2.2 Inter-phase and Project Level Dynamics

The phases of systems engineering process are tightly connected to each other, and therefore many feedback processes cut across more than one phase. Some of the coupling mechanisms across different phases of a system design that can lead to reinforcing processes are summarized in Figure 5. Here we have included only two phases for simplicity (conceptual and detailed design). Successive phases build on each other. Requirements are based on identified customer needs and inform a solution concept, the latter in turn impacts preliminary, conceptual and detailed design, and so on. The productivity and quality of work in each phase therefore directly depends the quality of work completed previous in phases. A solid and precise set of requirements can significantly help the solution identification process and save overall resources needed to complete the project. In fact the savings from doing a good job in the early phases are quite significant, as much as an order of magnitude in some domains (Boehm 1981). Therefore overall project performance (in terms of schedule, cost, and quality) depends on the quality of the work of the early phases. Poor needs assessment, requirements translation and definition, and conceptual design leads to many problems later on that trigger schedule pressure for all the remaining work and activates many intra-phase dynamics discussed (*R8: Error on Error*). Moreover, one of the hidden reactions to pressures is the relaxation of some of the testing criteria for acceptance of work in one phase. Such relaxation later hurts overall performance as it leads to the introduction of more defects into the work underlying the next phase (*R9: Cutting Tests*).

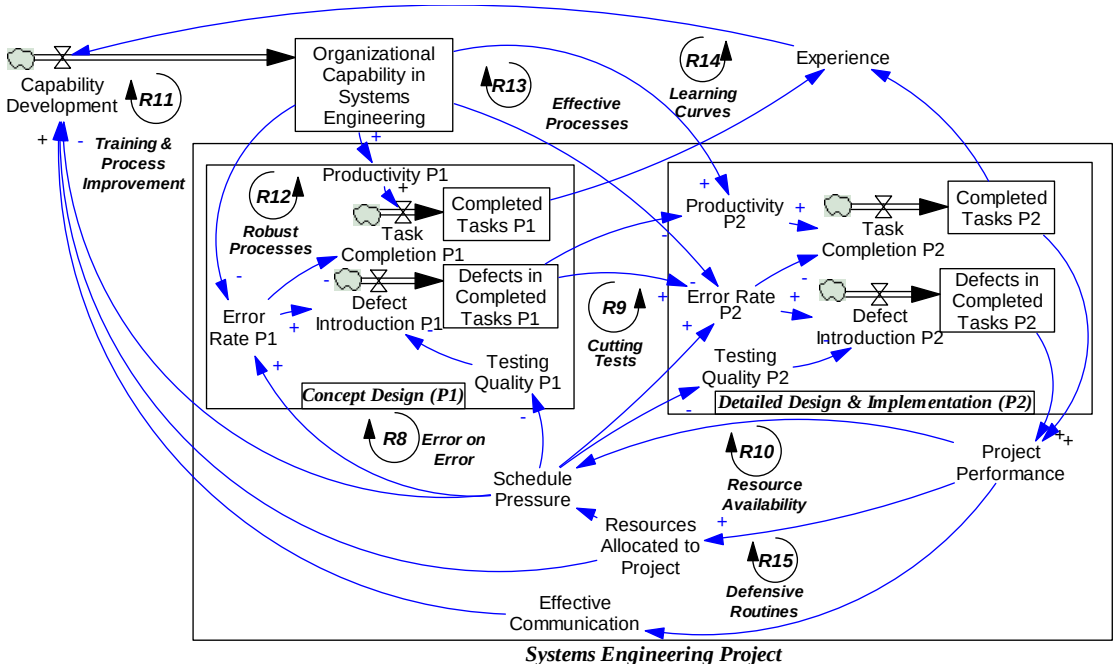

Figure 5: Inter-phase Dynamics

Project performance also often impacts the resources available for the continuation of the project. In the competition for limited enterprise resources, projects that have delivered according to their plans find themselves at a better bargaining position. Therefore in many enterprises a "success to successful" dynamic (Senge 1990) creates a link between project performance and schedule pressure in a project, closing the *R10: Resource Availability* reinforcing loop.

Enterprise capabilities are a central concept in the strategy literature (Grant 2002) and are helpful for understanding some other dynamics of interest in systems engineering projects. Capabilities are generally defined as routines and operating procedures (e.g., concurrent engineering, rapid prototyping, etc.) used to get the task done (Winter 2003). Systems engineering capability depends on the skills and experience of the team members, the robustness of the systems engineering processes used, the availability of automated tools and methods, among other. This capability directly impacts the quality and productivity of the systems engineering work conducted in an enterprise. Training, process improvement, and experiential learning are among the ways through which this capability can increase. These, in turn, depend on other endogenous factors: usually resource cuts first hit the long-term process improvement and training activities, leading to a potential long-term drop in enterprise capabilities (*R11: Training & Process Improvement*) while successful groups are more likely to benefit from such capability building opportunities (Sterman 2000). Projects with declining systems engineering capability level will suffer from lower quality and productivity, more trouble, and therefore will find fewer opportunities to fix their capability deficit (Loops *R12: Robust Processes* and *R13: Effective Processes*).

Learning curves also play an important, usually positive, role in longer projects. As different phases of a project progress, the teams involved learn from their experience and build more effective routines. They can therefore improve their productivity and quality and, everything else being equal, speed of the later stages of the work (*R14: Learning Curves*). These dynamics are shown to contribute to higher late stage quality/productivity in single projects (Lyneis, Cooper, and Els 2001), and to overall success of more experienced enterprises over longer times (Argote and Epple 1990). Finally, performance problems in projects lead to political dynamics inside the enterprise that activate defensive routines (Argyris 1985). Under stress and threats to their reputation, financial rewards, or job security enterprise members are more likely to reduce

effective communication in favor of saving their own position. Such defensive routines, however, negatively impact the capability of the enterprise to deal with the problems in the project at hand and solve them effectively. This completes another reinforcing loop, *R15: Defensive Routines*.

In summary, the dynamics that cut across multiple phases of a systems engineering project are based on the coupling of different phases through quality of successive phases, through the development and updating of requirements, the sharing of resources among different phases, and the enterprise capability in systems engineering used across them. These dynamics provide additional mechanisms that can explain the difference of successful and unsuccessful projects. In contrast to intra-phase dynamics, the loops discussed in this section act over longer time horizons, they start more slowly, but when under way, they are harder to stop. These dynamics also point to a different set of managerial levers. Specifically, members of teams involved in a single phase have little input in the allocation of enterprise resources. Strong managerial roles responsible across phases are required to manage these dynamics and make the necessary tradeoffs. Product development research has underlined the importance of such heavy-weight project managers (Wheelwright and Clark 1992).

3.2.3 Dynamics across Multiple Systems Engineering Projects

Most major enterprises, from defense agencies to private firms, are concurrently involved in several large scale systems engineering projects. The interactions across these projects lead to a third set of dynamics, summarized in Figure 6.

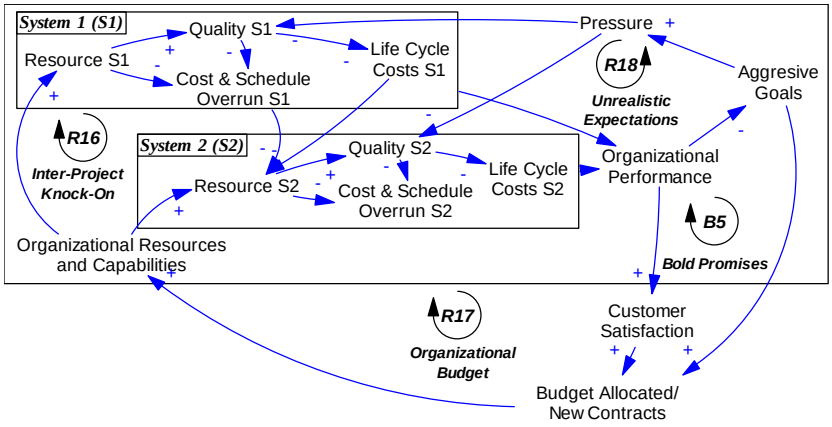

Figure 6: Dynamics across Multiple Projects

First, resources are often directly, or indirectly, shared across projects. Direct resource sharing include individuals who are involved in multiple projects at the same time, while indirect resource sharing is highlighted when one project can not receive all its desired resources because many key people are still engaged in another project that is running late. Resource sharing can lead to knock-on effects among different projects: activation of reinforcing loops in one project lead to delays and cost overruns, which in turn limits resource availability and timeliness for the next project. In fact, such knock-on effects are seen as more disruptive when we consider the fact that early stages of systems engineering projects are most critical to their success. Therefore early resource shortage often leads to low-quality concept development and design, which later disrupts the detailed solution development, implementation, and increases life cycle costs, leading to yet longer delays in the next projects (*R16: Inter-project Knock-On*). Such dynamics can move like a domino across projects and significantly reduce the capability of the enterprise to complete its projects on time and on budget (Repenning 2000; Repenning 2001).

Enterprise performance across projects significantly impacts customer satisfaction and therefore the propensity of winning new contracts (for commercial firms) or receiving the requested budget

(for governmental agencies). Budget and new contracts, on the other hand, directly influence available resources for successful completion of existing and new projects. Some firms utilize this reinforcing loop for their benefit and grow larger or more successful while others are hit hard by its negative ramifications (*R17: Organizational Budget*). Some enterprises try to avoid such decline by promising more aggressive goals to their customers, e.g. finishing the next more ambitious project with less resources (*B5: Bold Promises*). However, such aggressive goals are often counter-productive, as they raise the pressure level inside the enterprise beyond functional levels and lower the overall performance through higher error rates, poor testing, lower capability investments, etc (*R18: Unrealistic Expectations*).

The enterprise level, inter-project dynamics are most directly relevant to upper management and legislators. They relate to the management of a portfolio of systems engineering projects and defining the rules for participation in such projects. By understanding these dynamics, one can see how trying to do more actually can backfire and reduce the total output. It also suggests that enterprises may be better-off cancelling a fraction of their projects all together, rather than cutting the budget of every project by a similar fraction. However, this solution may not be feasible given the realities of the current acquisition process.

3.3 Feedback Loops and Exogenous Success Factors

In this paper we have focused on endogenous[2] feedback processes that impact systems engineering projects at multiple levels. This focus is helpful for identifying mechanisms that can be controlled or changed by managerial actions. Moreover, these feedback processes provide a framework for understanding the mechanisms through which many exogenous factors related to success and failure operate. We illustrate by two examples: technological novelty of a project and the contract awarding procedure.

Technological novelty is considered one of the prime causes of delays and cost overruns in projects (Tatikonda and Rosenthal 2000; Yetton, Martin et al. 2000). In our dynamic framework, technological novelty is seen as a factor that increases average error rate above typical projects since project teams need to learn many factors through trial and error. Higher base error rate, on the other hand, undermines initial project estimation (because error rate is unknown) and leads to optimistic estimates. These activate reinforcing loops through increasing schedule pressure, and therefore increase the chances of cost, budget, and quality problems. Furthermore, technological novelty leads to more uncertainty in terms of tasks involved and the amount of work that needs to be completed. In other words, it introduces more frequent and stronger unexpected changes during the project. As we discussed in Section 3.2.2, those unexpected changes are one of the main mechanisms for activation of vicious reinforcing loops. With this framework we can see more clearly through which mechanisms technological novelty can increase the risks of a project.

Contract awarding mechanisms that choose the bidder with lowest price negatively impact project performance. These contracting rules create an incentive for the bidders to estimate project costs optimistically. However, optimistic estimation leads to aggressive goals on productivity and schedule once the project is won. Aggressive goals increase schedule pressure, eat-up the safety margin associated with each project and compromises the project resiliency against unexpected changes, and thus increase the likelihood of dominance of dysfunctional reinforcing loops. Subsequent contract extensions and sub-awards cannot fix the main cause of the problem as long as the enterprise has passed the tipping point in terms of schedule pressure. Therefore, the enterprise does much worse than it potentially could.

[2] By endogenous we mean causes of success and failure that depend on project performance through time. Any factor determined independent of project performance or enterprise is considered exogenous.

4.0 Conclusions and Future Work

The feedback processes presented in the previous section provide the initial foundation for the integrated framework that would serve as the basis for understanding the dynamic complexity of the systems engineering process and offer the mechanism to test various assumptions about the design of systems engineering systems in particular. Nevertheless, there are a number of additional tasks that need to be accomplished. In terms of the inter phase dynamics, described in Section 3.2.2, they would have to be mapped in detail into the current structure of the systems engineering lifecycle process (Blanchard and Fabrycky 2006). This is very important given the dynamic nature of the requirements analysis (Clarke, Eisen, and Wyland 2008) and its contribution to potential system failures as depicted in Table 1.

Typically user needs are translated into system requirements through QFD or other means. Throughout each phase of the systems engineering process these requirements are further defined and allocated. Each time a requirement is analyzed or reconfigured it can change to the point that the original system requirement may not be allocated into the system with its original intent. The potential reasons why requirements change are many and can include temporal, political, fiscal, and/or cultural considerations. During each phase new issues may change the original intent of the requirement. For example, as noted in Section 3.2.2, budget constraints may render an initial user requirement too expensive to implement. Additionally one must consider that the temporal, political, fiscal, and cultural considerations can be responsible for a requirements creep into the system, diluting the original purpose of the system. This creates additional tasks that affect both the intra and inter process dynamics.

Another aspect that needs to be considered in the future, relates to one of the issues identified in Table 1, i.e., that systems designs are often plagued with inadequate cost estimation models. This is in part driven by the fact that cost estimating needs to be done early in the system's life cycle and often involves the consideration of new technologies. Enterprises must continually adopt and exploit new technologies to ensure that the systems they procure and use meet changing performance requirements and long-term cost goals. Unfortunately, adopting new technologies may bring unexpected consequences for the systems the enterprise procures, and for the provision of the necessary services required for the enterprise's long-term sustainability.

Nevertheless, the estimation of cost performance for any system requires that on the one hand analyst track the cost consequences of the dynamics represented in Section 3. On the other hand, one needs to understand and track the cost ramifications of new technology development (Monga and Triantis 2002), technology integration (Damle 2003), and systems operations, support and disposal (Scott 2003). One of the challenges that any analyst faces is that these cost estimation initiatives require cost data that may exceed the typical work breakdown approaches that have been used in the past or are currently being used today.

Another challenge is employing the wide-spread use of analytical tools within the systems engineering discipline. Systems engineers are often viewed as the owners of the systems processes, and nothing more. However, systems engineers bring a wide array of analytical tools that can help mitigate the problems addressed in the literature and in this paper. In this vane, one of our objectives is to define a unifying framework that will accommodate quantifiable systemic views of the entire systems engineering life-cycle process. Our research will include: (i) providing a comprehensive view of Model Based Systems Engineering (we find current definitions of Model Based Systems Engineering inadequate for application in engineering problems); (ii) defining how legacy, and potentially new, tools will be used for each phase of the life-cycle; and (iii) how these tools will be used together to support the systems engineering process. Some of our current work explores how to successfully use architecture based products (IDEF0, N^2, FFBD) to generate requirements. We then link architectures to modeling and simulation to test and refine many of the requirements and overall systems goals. Our linking of architectures to

requirements is a new endeavor, but we are encouraged by the findings thus far. We believe that analytical processes, models, and frameworks are what allow the Systems Engineer to look out counterparts in other engineering disciplines in the eye and say. “This is our answer as derived through analytical means.”

References

Abdelhamid, T. K. and S. E. Madnick (1983). "The Dynamics of Software Project Scheduling." Communications of the ACM, **26**(5): 340-346.

Argote, L. and D. Epple (1990). "Learning-Curves in Manufacturing." Science **247**(4945): 920-924.

Argyris, C. (1985). Strategy, Change, and Defensive Routines. Boston, Pitman.

Bar-Yam, Y. (2003). When Systems Engineering Fails – Toward Complex Systems Engineering. New England Complex Systems Institute.

Bahill, A. T. and S. J. Henderson (2005). "Requirements Development, Verification, and Validation Exhibited in Famous Failures." Systems Engineering, 8(1):1-14.

Blaisdell, J., Brazie., L., and K. Pelletier-Costa (2008). "Runway Incursions Elimination through a Cockpit Based Runway Incursion Prevention System. Systems Engineering Masters Project, Virginia Tech, Grado Department of Industrial and Systems Engineering.

Blanchard, B.S. and W. J. Fabrycky, W.J (2006). Systems Engineering and Analysis. 4th Edition, Pearson Prentice Hall.

Boehm, B. W. (1981). Software Engineering Economics. Englewood Cliffs, N.J., Prentice-Hall.

Boppana, K. et al. (2006). "Can Models Capture the Complexity of the Systems Engineering Process?" Engineering Systems Division, Massachusetts Institute of Technology, and Wojcik, L.A., Center for Advanced Aviation System Development, The MITRE Corporation.

Briggs, C., and P. Little (2008). "Impacts of Organizational Culture and Personality on Decision-Making in Technical Organizations." Systems Engineering, 11(1): 15-26.

Brooks, F. P. (1995). The Mythical Man-Month : Essays on Software Engineering. Reading, Mass., Addison-Wesley Pub. Company.

Clarke, K., Eisen, B., and M. Wyland (2008). "System Engineering and Integration: A Traffic Information System." Systems Engineering Masters Project, Virginia Tech, Grado Department of Industrial and Systems Engineering.

Cooper, K. G. (1980). "Naval Ship Production: A Claim Settled and a Framework Built." Interfaces **10**(6).

Cooper, K. G. (1994). "The $2,000 Hour: How Managers Influence Project Performance through the Rework Cycle." Project Management Journal **15**(1): 11-124.

Damle, P. (2003). "The Implementation of New Technologies on Ship Systems: A System Dynamics Approach for the Integration of New Technologies." MS Thesis. Virginia Tech Grado Department of Industrial and Systems Engineering.

Eriksson, M., Borg, K, and J. Borstler (2008). "Use Cases for Systems Engineering – An Approach and Empirical Evaluation." Systems Engineering, 11(1): 39-60.

Ford, D. N. and J. D. Sterman (2003). "Overcoming the 90% Syndrome: Iteration Management in Concurrent Development Projects." Concurrent Engineering-Research and Applications **11**(3): 177-186.

Graham, A. (2000). "Beyond PM 101: Lessons for Managing Large Development Problems." Project Management Journal **31**(4): 7-18.

Grant, R. M. (2002). Contemporary Strategy Analysis : Concepts, Techniques, Applications. Malden, Mass., Blackwell Business.

Hari, A., Kasser, J. E., and M. P. Weiss (2007). "How Lessons Learned from Using QFD Led to the Evolution of a Process for Creating Quality Requirements for Complex Systems." Systems Engineering.10(1): 45-63.

Haskins, C. (2007). "Using Patterns to Transition Systems Engineering from a Technological to Social Context." Systems Engineering 10, 1-9.

Johnsrud, L. K., R. H. Heck and V. J. Rosser (2000). "Morale Matters - Midlevel Administrators and their Intent to Leave." Journal of Higher Education **71**(1): 34-+.

Kossiakoff, A. and W. N. Sweet (2003). Systems Engineering Principles and Practice. John Wiley and Sons, Inc., Hoboken, NJ.

Lewis, J. P. (2007). The Project Manager's Desk Reference. 3rd edition, McGraw-Hill, New York, NY.
Lyneis, J. M., K. G. Cooper and S. A. Els (2001). "Strategic Management of Complex Projects: A Case Study using System Dynamics." System Dynamics Review **17**(3): 237-260.
Lyneis, J. M. and D. N. Ford (2007). "System Dynamics Applied to Project Management: A Survey, Assessment, and Directions for Future Research." System Dynamics Review **23**(2-3): 157-189.
McConnell, S. (1988). Software Project Survival Guide. Microsoft Press, Redmond, WA, 1998.
McLucas, A.C. and M. J. Ryan (2005). "Meeting Critical Real-World Challenges in Modeling Complexity: What System Dynamics Modeling Might Learn from Systems Engineering." University College, University of New South Wales, Australian Defense Force Academy,
Monga, P. and K. Triantis (2002). "The Behavior of New Technology Development: A System Dynamics Approach." Twentieth International Conference of System Dynamics Society. Palermo, Italy, August 2002.
Nepal, M. P., M. Park and B. Son (2006). "Effects of Schedule Pressure on Construction Performance." Journal of Construction Engineering and Management-ASCE **132**(2): 182-188.
Nowinski, E.H. and R. J. Kohler (2007)."The Lost Art of Program Management in the Intelligence Community." http://www.cia.gov/csi/studies/vol50no2/html_files/Program_Management_4.htm.
Rahmandad, H. (2005). Three Essays on Modeling Dynamic Organizational Processes. Sloan School of Management. Cambridge, Massachusetts Institute of Technology. Ph.D. Dissertation.
Rahmandad, H. and D. Weiss (2008). "Dynamics of Concurrent Software Development." System Dynamics Review **Under Review**.
Repenning, N. P. (2000). "A Dynamic Model of Resource Allocation in Multi-project Research and Development Systems." System Dynamics Review **16**(3): 173-212.
Repenning, N. P. (2001). "Understanding Fire Fighting in New product Development." The Journal of Product Innovation Management **18**: 285-300.
Rudolph, J. W. and N. P. Repenning (2002). "Disaster Dynamics: Understanding the Role of Quantity in Organizational Collapse." Administrative Science Quarterly **47**: 1-30.
Sage, A.P. and S. M. Biemer (2007). "Processes for System Family Architecting, Design, and Integration." IEEE Systems Journal, 1(1).
Scott, J. (2002). "A System Dynamics Model of the Operations, Maintenance and Disposal Costs of New Technologies for Ship Systems." MS Thesis. Virginia Tech Grado Departmen t of Industrial and Systems Engineering.
Senge, P. M. (1990). The Fifth Discipline: The Art and Practice of The Learning Organization. New York, Currency Doubleday.
Sobrero, M. and E. B. Roberts (2001). "The Trade-Off between Efficiency and Learning in Inter-organizational Relationships for Product Development." Management Science **47**(4): 493-511.
Smith, A. E. (October 22, 2007). "Where's the Leadership?" *Space News*, p. 19.
Sterman, J. (2000). Business Dynamics: Systems Thinking and Modeling for a Complex World. Irwin, McGraw-Hill.
Tatikonda, M. V. and S. R. Rosenthal (2000). "Technology Novelty, Project Complexity, and Product Development Project Execution Success: A Deeper Look at Task Uncertainty in Product Innovation." IEEE Transactions on Engineering Management **47**(1): 74-87.
Taubman, P. (2007). "In Death of Spy Satellite Program, Lofty Plans and Unrealistic Bids." New York Times, November, 2007.
Taylor, T. and D. N. Ford (2006). "Tipping Point Failure and Robustness in Single Development Projects." System Dynamics Review **22**(1): 51-71.
Weakliem, D. L. and S. J. Frenkel (2006). "Morale and Workplace Performance." Work and Occupations **33**(3): 335-361.
Wheelwright, S. C. and K. B. Clark (1992). Revolutionizing Product Development : Quantum Leaps in Speed, Efficiency, and Quality. New York, Free Press.

Winter, S. G. (2003). "Understanding Dynamic Capabilities." Strategic Management Journal **24**(10): 991-995.

Winter, D. C. (2007). "Getting Shipbuilding Right." Naval Institute Proceedings, June, pp. 16-20.

Wojcik, L.A. and K. C. Hoffman (2006). "Systems of Systems Engineering in the Enterprise Context: a Unifying Framework for Dynamics." Center for Advanced Aviation System Development and Center for Enterprise Modernization, the MITRE Corporation, IEEE, 24-26 April 2006.

Yetton, P., A. Martin, R. Sharma and K. Johnston (2000). "A Model of Information Systems Development Project Performance." Information Systems Journal **10**(4): 263-289.