Comments Welcome

Program Architecture and Adaptation

Tyson R. Browning
Neeley School of Business
Texas Christian University
TCU Box 298530
Fort Worth, TX 76129 USA
T.Browning@tcu.edu
www.TysonBrowning.com

Prepared for the

Symposium on Complex Systems Engineering

Hosted by the RAND Corporation
Santa Monica, CA
January 11-12, 2007

Keywords: Project modeling, project architecture, process modeling, organization modeling, systems architecting, emergence, adaptability

Acknowledgment: The research assistance of US Army Capt. Marc Ortiz on the case studies is greatly appreciated.

1. Introduction

The intent of this paper is to prompt further discussion. The ideas herein are preliminary. While this paper will use the term "program," meaning a group of interdependent projects[1], its ideas also apply to the higher, enterprise level and the lower, project level. Examples of programs include the development of large, complex product systems such as aircraft or spacecraft.

Programs are extremely complex entities. They combine the challenges of engineering product systems with those of managing people, organizations, tools, processes, schedules, and budgets. As supply chains, partnering arrangements, outsourcing, globalization, technologies, capabilities, and stakeholder desires have grown in size and sophistication, program planning and control has become even more difficult. As a result, programs are notoriously challenged to deliver desired outcomes—i.e., a result with pre-specified levels of performance by a deadline and within a budget. The working assumption in this paper is that program managers do not have adequate decision-support systems and are faced with information overload. They are often surprised by emergent problems in programs. However, it is interesting to note that such surprises to the program manager are often known much earlier by *someone* on the program. No matter how good the "problem discovery" capabilities on a program, they are of limited use unless accompanied by effective "problem transmission" capabilities.[2] Addressing these issues would seem to require the recognition and treatment of programs as complex systems—systems that perhaps can be engineered in a better way, or at least modeled and better understood.

While the product (or service) to be developed by a program has long been the focus of systems engineering, this is just one of several important systems in a program. The process though which development occurs, the organization that executes this process, the tools they use, and the goals they seek are each an additional system of elements with relationships. There are innumerable examples of how optimizing any one of these systems can sub-optimize the others. Therefore, there is a need to

[1] A project is "a temporary endeavor undertaken to create a unique product, service, or result" (PMI 2004).
[2] This problem is a microcosm of the same problems faced by large governments trying to prevent terrorism, for example. That is, it often turns out that someone knew about each terrorist event before it occurred, but this information could not be transmitted so as to penetrate the filters surrounding the decision makers in a timely manner.

examine the systems in relation to each other and more holistically.

Models have long been used to improve understanding and visibility of complex systems. Models are an abstraction of reality. As Box famously stated, "All models are wrong, but some are useful." This paper proposes a framework in which program managers or their designees can build models that help them to visualize, understand, analyze, improve, and manage the systems, interactions, and emergent behaviors in programs.

The paper consists of two main parts. The first discusses an approach to modeling five of the systems in a program and their interactions. The second discusses adaptation and emergence in the context of programs and the five systems, drawing examples from four case studies.

2. Part I: A Program Architecture Modeling Framework

2.1 Five Program Systems

A program consists of at least five systems: the product, the process, the organization, the tools, and the goals (Figure 1). Each of the five systems is related to the others, is composed of elements with relationships, and thus can be discussed in terms of its network structure and architecture. The product system is the desired result of the project—i.e., in the case of product development, the product "recipe." Here, the system consists of designed components (hardware, software, and/or people) which may be related via a variety of types and degrees of interactions. The hierarchical relationships in the product system are often represented with a *product breakdown structure* (PBS). The process system is the work done and interim results achieved to produce the product system. The process system consists of activities related to each other by a flow of work products or deliverables. While the vertical relationships among activities are described by a *work breakdown structure* (WBS), the corresponding decomposition of deliverables is often ignored, unfortunately. The organization system consists of people assigned to do the work to produce the product system—i.e., individuals, groups, teams, or other organizational units, related to each other by communication, reporting, etc. The hierarchical relationships in this system are represented in the *organization breakdown structure* (OBS), commonly seen as the ubiquitous

organizational chart. The tool system represents the technologies used by the people (in the organization) to do the work (in the process) to get the product. While some tools, like drafting boards and pencils, may be essentially unrelated, many contemporary software tools must interact as a direct result of the relationships between activities and the people doing them. The software applications and information technologies (IT) used by an organization to accomplish work constitute a significant portion of the tool system. The product, process, organization, and tool systems operate in the context of requirements or goals, which may themselves be related: e.g., making it easier to meet one goal may make it more difficult to meet another. Hence, a fifth system is the goal system. Some of the hierarchical relationships in the goal system have been addressed in the context of requirements flow-down. Each of the five systems is related to and both enables and constrains the others, even though Figure 1 does not show all such relationships. An enterprise typically has multiple ongoing programs (represented by the layers in Figure 1), so there are strong incentives to achieve commonality in these five systems across programs. Browning *et al.* (2006) provide additional discussion of the history of this line of thought.

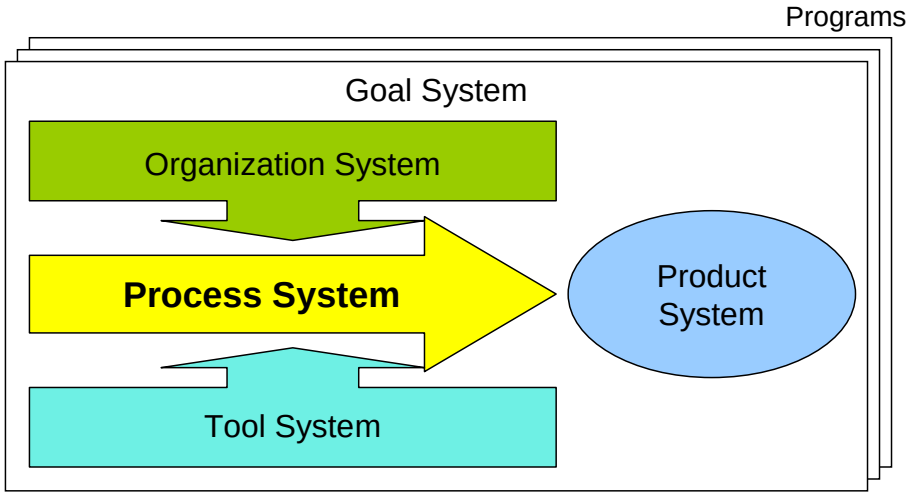

Figure 1: Five Systems in a Project or Program (Browning *et al.* 2006)

Each of the five systems has a structure or architecture, characterized by its constituent elements

and their internal and external relationships that give rise to its behavior over time.[3] While some think of this architecture primarily terms of the vertical, hierarchical relationships between elements—formed through the decomposition of the system into simpler elements—it seems clear upon further reflection that the lateral relationships among elements play the greater role in the behavior of the system. It is the difference between having the millions of parts used to build an aircraft fitted together as a working product versus having them organized in bins according to the PBS.

Figure 2 provides another way to think of the five systems, as a "capability molecule." That is, the capability to accomplish something is incomplete without the right combination of integrated people, processes, and tools focused on producing the right components of a desired product or result. Just as the properties of a molecule differ from those of its individual elements, so do the properties of a program differ from those of its individual product, process, organization, tool, and goal systems. Because programs are unique, using the same process, organization, or tools on multiple programs does not guarantee that each will have the same outcome. Some of this variance comes from exogenous effects, while some comes from different interactions among the systems.

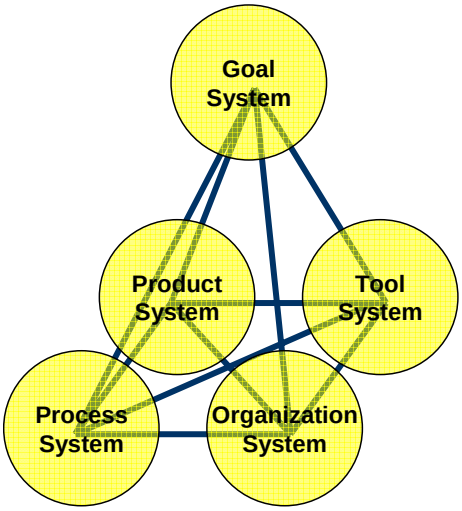

Figure 2: A "Capability Molecule"

[3] This is a paraphrase of the definition of a product architecture in IEEE STD 610.12: "The structure of components, their relationships, and the principles and guidelines governing their design and evolution over time."

Because the five systems are interrelated, and because the optimal program may not consist of the optimal instances of each of the five systems, it seems interesting and potentially rewarding to consider the five systems together. I refer to the integration of the five systems as the program system and to its structure (in terms of its constituent elements and their relationships) as the *program architecture*.

2.2 Modeling the Program Architecture

A variety of models have been proposed for each of the five systems independently. For example, the product architecture literature (e.g., Alexander 1964; Fixson 2005; Rechtin 1991; Sharman and Yassine 2004; Ulrich 1995) discusses product decomposition into components and their relationships. The program process modeling literature (Browning and Ramasesh 2007) has also recently begun to discuss processes in terms of their architecture (Browning and Eppinger 2002). Some organization design literature (e.g., Allen 1977; Browning 1999; Galbraith 1994; Galbraith *et al.* 1993; Lorsch 1982; Lorsch and Lawrence 1972; Mintzberg 1979; Tushman and Nadler 1978) conceives of organizational units interrelated by flows of information and other objects. In addition, some literature addresses more than one of these systems at a time in an effort to explore their relationships (e.g., in the case of the product and organization systems, Sanchez and Mahoney 1996; Sosa *et al.* 2002; Sosa *et al.* 2003). The point of this paper is not to provide a comprehensive review of these streams of literature.

Some recent papers have begun to conceive of the program architecture (without using that term) as three or more of the systems (Bartolomei *et al.* 2006; Browning 2001; Eppinger and Salminen 2001; Negele *et al.* 1997; Strandberg *et al.* 2006). Most of these papers use the *design structure matrix* (DSM) as the means for modeling an individual system. A DSM is a square matrix that maps the elements in a system to each other (like an N^2 diagram), providing two main advantages: concise representation and integration analysis (Browning 2001). One reads across a row of the matrix to see where the element in that row sends its outputs, and one reads down a column to see from where the element in that column receives its inputs. For example, in the DSM in Figure 3, element 7 provides an output to elements 5, 6, 8, and 13 (reading across row 7) and receives an input from element 1 (reading down column 7). In a product system model, the DSM elements represent components; in a process system model, they

represent activities; and in an organization system model, they represent organizational units. Thus, a DSM is a generic system modeling tool that can be adapted to different kinds of systems. DSM models utilize Simon's (1962) concepts of hierarchy and decomposition.

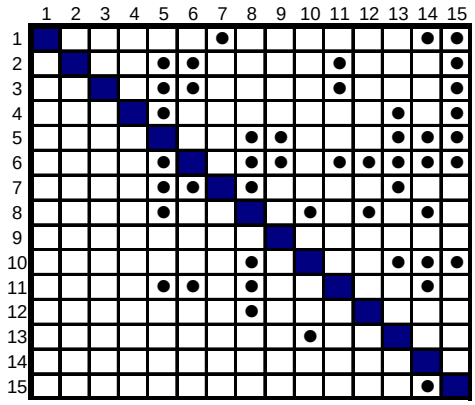

Figure 3: Example DSM

Danilovic and Browning (2007) standardized the term *domain mapping matrix* (DMM) for any rectangular matrix that maps between the elements of any two project systems or domains. For example, role-responsibility matrices have long been used by project managers to assign people and teams to activities, which is a mapping from the organization system to the process system. While some such inter-domain mappings are common, some had not yet been discovered. Therefore, Danilovic and Browning proposed the grouping of DSMs and DMMs shown in Figure 4. The order of the five DSMs in the figure is arbitrary; future research might enable prescription of a preferred order. By synchronizing information about these five systems and their interactions in a program, we have an improved basis for representing and analyzing the sources of complexity, uncertainty, and dynamism in programs. Since each of these DSMs and DMMs represents an important "element" of a program, we refer to Figure 4 as a kind of "periodic table," since it reminds us of the classical periodic system of physical atomic elements. Much as the periodic classification of the elements spurred the investigation and discovery of previously unknown elements to fill in the gaps in the table, Figure 4 enables us to hypothesize the presence of potentially enlightening inter-domain representations and analyses in programs.

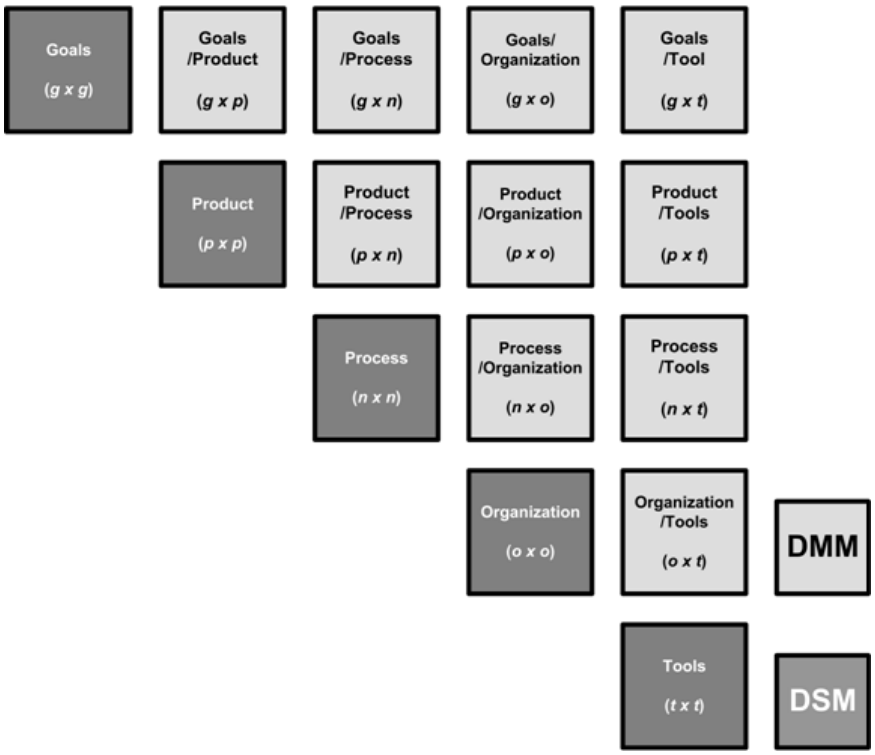

Figure 4: A "Periodic Table" of Elements Comprising the Program Architecture

My current research is building a set of research questions around this framework and building models of case study programs' architectures. Explorations of each of the individual domains has led to numerous and helpful insights for managers. Exploring them all of once promises to be rewarding yet is also quite challenging. I am very interested in thoughts, comments, and reactions to this line of thinking and modeling, which seems to hold promise for helping program managers organize, integrate, analyze, and benefit from the vast amounts of information in programs. The next part of this paper looks at one stream of program architecture research questions in particular.

3. Part II: Program Adaptation and Emergence

With a rich model of program architecture, one would indeed have an advantageous vantage point for program planning and control. However, since each program is unique, some might argue that programs cannot really be designed; rather, they emerge. Both points of view are not mutually exclusive, and perhaps both contain elements of truth. Maybe both also contain some misleading ideas, however. This tension shows up in many areas, such as in the use of standard processes, formal organization

structures, product platforms, and standard software tools. How much of the program structure can or should be specified *a priori*, and how much should be left to the discretion of the program participants? How much should a program standardize roles, responsibilities, personnel reporting relationships, processes, and tools? Is agility, flexibility, or adaptability the opposite of rigidity and standard structure, or is it enabled by it? Giving one of our tentative answers up front, we find that *the imposition of appropriate structures is an enabler of adaptability, to a point, and then becomes detrimental to it*. Therefore, we are interested in how to identify this point and how to determine the proximity of any program to it. We are also interested in whether the "optimal" (most valuable) product, process, organization, tool, and goal structures (architectures) are designed directly, or rather if they must be designed indirectly, by only designating the policies and rules whereby the program participants (agents) will interact and evolve these five structures to their highest-value levels. To begin exploring these questions, this section of the paper develops a framework for thinking about program structure, adaptability, and value and explores two sets of coupled case studies.

3.1 Formality, Standardization, Adaptability, and Agility

Formality and standardization are defined as "the establishment of routines or rules which constrain [the] action[s] of each unit" (Thompson 1967). Currently there are no strict quantitative means of measuring formality and standardization. We use three, broad categories: formal, semi-formal, and informal. An example of a semi-formal enterprise is Toyota. Its "rigid specification is the very thing that makes the flexibility and creativity possible" (Spear and Bowen 1999). However, too much standardization may preclude further experimentation and discovery: an enterprise continually uses old processes that are not producing at an optimal value and does not seek newer processes (Levitt and March 1988).

Adaptability is used in a variety of business discussions as a synonym of flexibility, agility, and responsiveness (Haeckel 1999). Agility is defined as "the ability to manage and apply knowledge effectively, so that an organization has the potential to thrive in a continuously changing and unpredictable business environment" (Dove 2001). This is the definition we will use for adaptability as

well. Adaptability is a vital characteristic of a successful enterprise or program. Adaptability allows a program to take the necessary actions needed to compete and survive in its operating environment. Indicators of an adaptable program include flat (lateral) interaction, distributed control and information, plug compatibility, elastic capacity, redundancy, and diversity (Dove 2001).

3.2 The Program Emergence Cycle

Emergence is defined as the concept that systems are dynamic and changing over time, due to the simple rules that generate complexity (Holland 1998). What polices or rules facilitate the emergence of a higher-value process or program? Like living organisms, programs must adapt as necessary to changing environmental conditions to be successful. I suggest contemplating this ongoing adaptation in terms of a *program emergence cycle*, shown in Figure 5, which is based on a closed-loop design akin to a cybernetic controller. A program's desired goals and outcomes are compared to actual ones; any difference implies a value gap.[4] A value gap provides the stimulus for change, if the gap is recognized. Programs that recognize gaps more quickly will move faster through the emergence cycle. According to Nagle, "the institutional learning process begins with the recognition of shortcomings in organizational knowledge or performance" (Nagel 2002). Programs can also differ in their tolerance for value deviations before seeking change. A higher tolerance delays the emergence cycle. The desire to change prompts change actions, which can be applied to one or more of the five program systems directly or indirectly. For example, the program may attempt to change its product architecture, its processes, its organization structure, its tools, or even renegotiate its requirements and goals. A change in one system may purposely or inadvertently cause a change in another system. A program's ability to change these systems and the rate at which changes are possible both regulate the speed of the change process; both are often affected by the degree of standardization and formality in the program's systems. Short, tight interaction and feedback loops in all project systems facilitate adaptability. Once changes are effected, a new outcome will emerge, and the cycle repeats.

[4] "Specifying outcomes, rather than the activities undertaken to produce them, is the essence of adaptive system design" (Haeckel 1999).

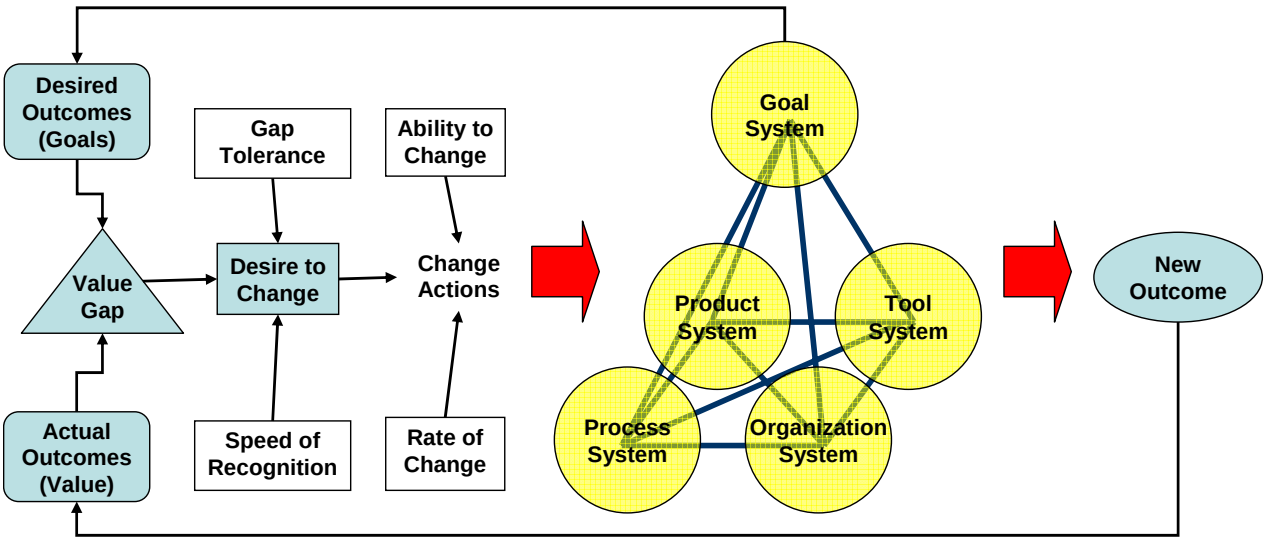

Figure 5: A Program Emergence Cycle

In addition to having the right attitudes towards change, moving through the cycle quickly requires that program managers have access to information about their five systems and the interactions among them. This information must be verified and valid across all of the systems—i.e., the process model cannot be making different assumptions than the goal system or the tool system. This requires an integrated modeling framework like the one presented in the previous section.

3.3 Program Value as a Function of Standard Structures

We will consider program value as a function of the benefits provided and sacrifices required of the program's stakeholders. Rather than exposing a mathematical formulation of program value (which I am working on elsewhere), we will use the idea of a valuable program being one that can provide the greatest benefits to the most important stakeholders for the least sacrifices. Usually the stakeholders (or a subset of them) will identify a set of goals for a program, in terms of time, cost, and quality, and the program that can muster the best integration of people, processes, tools, and product design will add the most value by meeting these goals the most efficiently, effectively, and predictably.

At one extreme, a group of disparate, unstructured people and tools can find it greatly difficult to

create value. The absence of communication protocols, shared knowledge, compatible tools, and common goals renders the timely and economical accomplishment of goals quite problematic. At the other extreme, a group of rigidly structured people, processes, and tools can also find it challenging to accomplish anything except perhaps an anachronistic goal, and even that with great inefficiency. The mandated use of fixed communication channels and protocols, cumbersome procedures, limited tools that work together only in specific ways, and shared mental models best characterized as "group think" can also impede the creation of value. "Formal structure handles routine extremely well" but does not "deal with the novel and unexpected events associated with change" (Kotter 1990). Between these two extremes of structure, each of which provides little value, I hypothesize the existence of a point where a certain degree of structure (or combination of structure and flexibility) provides maximum value. This point will undoubtedly vary depending on the particular goals sought by a program.

See Figure 6 for an attempt to diagram these ideas. The *x*-axis represents the degree of structure and rigidity in a program—in terms of people, products, processes, tools, and goals—and the *y*-axis represents the value provided by the program. The plotted curves are hypothetical value curves. The hypothesis is that the greatest value will be obtained from a program between the extremes. Various value curves such as 1, 2, and 3 may exist and provide different maximum values. As a program moves to the left or right on the *x*-axis, it will either increase or decrease the value of its results.

As examples of programs towards each extreme, we contrast two "programs" involved in Operation Iraqi Freedom in the 2003-2004 time frame, a U.S. Army unit and the Sunni-based insurgency. The insurgency began as many disconnect cells of people with a variety of goals, procedures, and tools. They had very little except desire and flexibility. Over time, the cells began to work together, sharing knowledge and tools and procedures, developing communication links and protocols, and coordinating attacks. While such cooperation limited their flexibility, the value (to them) of their results improved. Similarly, the U.S. Army found that its initial procedures, personnel relationships, and tools were not providing the desired value either. Giving more autonomy and decision control to local commanders was part of a response that reduced rigidity and led to improved results. While neither program has achieved

maximum value or full success (and the conflict continues), it is interesting that the less-structured program moved in the direction of increased structure, and the more-structured program moved in the direction of decreased structure, both with the intent of increasing the value of their results.

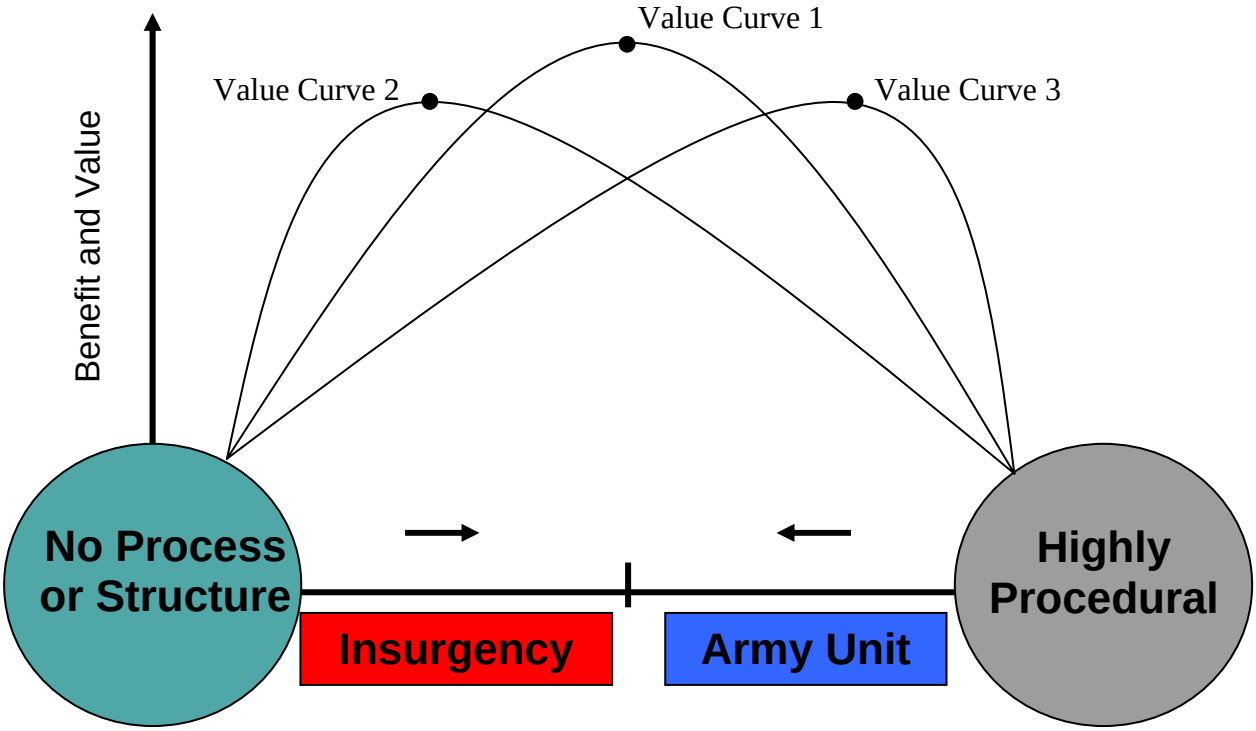

Figure 6: A Framework for Considering Program Value as a Function of Standard Structures

This framework prompts some general questions regarding program (and enterprise and organizational) change: What drives a change in the level of program structure and flexibility? What governs the rate of change? What rules or policies and other characteristics govern the path and rate of change? How do these factors apply to unique programs, in contrast to enterprises with ongoing operations? What are the specific ways in which structure aids and inhibits value? Paradoxically to some, the Toyota Production System is both rigidly defined and highly flexible (Spear and Bowen 1999).

3.4 Applying the Framework: Four Case Studies

To begin exploring these questions, we consider four case studies in two sets of two. In the first

couplet, we look further at the US Army and the insurgency in Iraq. In the second couplet, we look at two rough analogues for formality and informality in the business world, programs at Microsoft and Google. Interestingly, we find general support for the hypothesis of programs providing insufficient value with either too much or too little formal structure. Prior research has used the case analysis approach to determine how a process adapts during a particular scenario (Hutchins 1991). Other papers have studied organizations and changing routines, such as in a student housing department (Feldman 2000). One paper analyzed three different companies to determine how change develops and is implemented (Staudenmayer *et al.* 2002).

3.4.1 First Case Couplet: The Army and the Insurgency in Iraq

(A) Route Irish Operations

As an enterprise, the US Army can generally be positioned on the far right of the *x*-axis in Figure 6. The Army documents all of its processes in field manuals, standard operating procedures (SOPs), memoranda, and administration publications. The Army follows a set of routines that can be defined as "repeated patterns of behavior that are bound by rules and customs and that do not change very much from one iteration to another" (Feldman 2000). Changing Army processes is time-consuming, costly, and usually directed from the top. Without trying to analyze the Army holistically, we examine a specific mission requirement supporting the Army's overall goals in Iraq. We therefore look at a specific program, a specific goal given to a specific Army organizational unit.

The Army's goal in Iraq is to establish "a free, democratic, and secure Iraq" (Chiarelli 2006). A critical aspect of this goal is the safety and security along a strategic route connecting Baghdad International Airport to downtown Baghdad. This route is called Route Irish by all coalition forces. We look at one unit's attempt to provide security on this route despite having the additional mission of securing a sector south of Route Irish. Securing a sector consists of constant patrols, coordinating reconstruction projects, and interacting with the local officials. Securing a major strategic route requires major combat power and a dedicated combat unit.

The sector assigned to this unit was the al Rashid district of Baghdad, whose area is

approximately 60 square kilometers. Route Irish forms the northern border of this sector and the unit also is responsible for this route. During the months of June and July 2004, attacks against supply convoys along the route began to increase in frequency. Most of these attacks were hit and run attacks which amounted to little or no battle damage on the coalition forces. Higher headquarters continued to give the route little direct attention, despite its known strategic importance.

In the last week of July 2004, a security contractor convoy was hit with a car bomb, which garnered publicity in the national media. The question became "How safe is Baghdad, if security convoys escorting VIPs can't even travel from the airport to downtown safely?" The message underscored the strategic and political importance of security on Route Irish. Higher headquarters ordered Route Irish be secured by the unit responsible for the route. Higher headquarters expected the route to be secured prior to the incident, but it was never a priority. The value gap was created by an obvious increase in attacks and their effectiveness (due to the insurgency's adaptation) and the awakening perception among the higher-level commanders that "Route Irish was not safe." In the presence of this significant value gap, the unit was ordered to make changes.

The insurgents set car bombs in broken-down vehicles, they used gasoline canisters and debris to conceal improvised explosive devices (IEDs), and they buried IEDs in unpaved portions of the highway. This prompted US Army process changes such as: removing all vehicles stopped along the highway, removing all black market gasoline vendors, and only driving on paved roadways. The ideas of and mistakes made by the people in a program serve to stimulate change with alacrity (Feldman 2000).

The formality of the Army's structure had contributed the value gap. The Army's formal structure hindered its ability to identify the gap quickly. Route Irish's strategic importance was known throughout the Army command structure, but a unit was never formally tasked with providing security to the route. The timing of the Route Irish attack was critical to initiating change. "The timing of events has been found to enforce routines, focus energies and attention, shape how people approach tasks, and give meaning to actions and events" (Staudenmayer *et al.* 2002). In addition, "many observers agree … that a crisis is often needed for large-scale change" to occur (Else 2004). Unfortunately, "organizational

learning, when it does occur, tends to happen only in the wake of a particularly unpleasant or unproductive event" (Nagel 2002). As a program or enterprise grows in size and formality, it becomes harder for it to recognize pending threats and value gaps, and for it to take action to counter these threats or fill these gaps. The Army focused on Route Irish only after a serious attack received media attention. And while the Army continues to adapt to the tactics, techniques, and procedures of the insurgency, it has not necessarily adapted to putting an insurgency to rest or preventing one from rising.

Table 1 provides an overview of adaptations in the five systems of the Route Irish program. Several of these adaptations served to move the program a bit to the left of its original position on the *x*-axis in Figure 6.

Table 1: Route Irish Program Adaptations

	Before	Adaptation	After
Goals	Maintain stability in assigned sector, including Route Irish	Increased patrol presence, 24 hour operations on Route Irish	Maintain security and presence on Route Irish in particular
Product or Service	Relative stability within sector, but increasing attacks on Route Irish	Refined lines of operations, increase in patrols and presence	Attacks decreased immediately
Process	Maintain presence; set up random traffic control points; counter-mortar patrols; Route Irish presence part of natural patrol route	Increased combat power on route	Removing stopped vehicles; almost 90% of combat power on Route Irish; continuous patrol presence on route
Organization	Hierarchal, Vertical	Increased information sharing and flow	Increased network connectivity to troop command posts; improved information flow
Tools	Bradley Fighting Vehicles, M1 Abrams, Up-Armored HUMVEES	Improved systems to defeat IEDs	Increased capabilities, Warlock systems

(B) Insurgent Operations

The insurgents' goals in Iraq are as varied as the number of different insurgent groups. While many of these groups had sub-goals and ulterior motives for their actions, the goal of most of the active groups during the 2003-2004 period was to evict the US military from Iraq. We locate the insurgent

groups on the left side of the *x*-axis in Figure 6. This placement is based on their *ad hoc* methods of formulating and executing their attacks. The insurgents do not have formal training for the most part and they develop their processes through experience. In this analysis, we will also focus on certain groups. We recognize that we do not have complete information on the insurgents and their activities.

The organization of a typical insurgent group was very loose-knit and cellular in composition. It might not even have a formal leader or hierarchy. Each group has its own motives and sub-goals. Over time, various groups recognized that they shared common goals. Thus began the transformation of a disconnected, uncoordinated insurgency into a networked and coordinating group of insurgents. The emergence of a more formal enterprise allowed them to improve their processes and share information on ways to boost their impact and adaptations to US Army tactics.

This critical development of sharing information significantly improved the insurgents' processes. They shared information through the Internet and DVDs. This information consisted of the latest tactics and bomb-making techniques. We can assume that information on pressure-activated bombs and shape charges was sent in this manner (Jervis 2005). Time-sensitive information was relayed through cell phones to other insurgents to coordinate attacks.

A great example of the effects of this increased coordination is given by an insurgent that reconnoiters, plans, and directs suicide bombings (Ghosh 2005). He is a middle-man or handler between various insurgent groups and the attack itself. This insurgent states that he does not care about the sub-goals of the insurgent group that hires his services. He knows that the overall goal of removing the US military is the main goal, and this is his motivation. This process starts when an insurgent group contacts him and provides a person to carry out the attack. The handler then provides the explosives and plans and identifies the target. He will then assist the bomber to make sure they know the area and the surroundings. Eventually the handler will take him to the location for the attack (Ghosh 2005).

The previous example clearly represents a transformation in the insurgency. The handler stated that initially other insurgents staged their own attacks, but they were largely ineffective due to the superiority of the US Army's weapons. The handler then began to receive contacts from other insurgent

groups (Ghosh 2005). At this point the insurgents began to share information and resources. Once this network began to develop, the insurgent processes for attacks began to improve significantly. The development of an informal network of insurgents was critical to adaptation, since it "can deal with the greater coordinating demands associated with non-routine activities and change" (Kotter 1990).

How did the insurgency develop some lethal tactics in a very short amount of time, as compared to the US Army's lag in overall adaptation? Several reasons exist for the rapid evolution in the insurgents' processes. The organization of the insurgents is extremely loose and is not bounded by a conventional structure. The survival of the insurgency depends on the visibility and effectiveness of its attacks, which depend on the improvement of its processes. The various insurgents recognized they shared similar goals. The rapid sharing of information facilitated the actions necessary to reach that goal.

Going back to our Route Irish example, the unit that provided security on the route carried its mission out successfully as the numbers of attacks were reduced significantly. A major reason for this reduction was the amount of heavy armor placed on the route. By the end of the summer of 2004, the Route Irish security mission was given to another unit. This unit did not have the armor capabilities of the previous unit. The insurgency recognized this fact and almost immediately there was a significant increase in car bomb attacks on the route. We can only surmise the process adaptation that occurred, but it is safe to assume the insurgents recognized that a less-armored unit took over the mission. The insurgents then carried on with their attacks. They simply adapted to the situation and postponed their attacks until the right moment. This lends evidence to the fact that supporting groups were passing information about US forces. These supporting groups and handlers made direct coordination to conduct the attacks. In the words of a US Army tank commander, "They adapt to us and we adapt to them; it's a never-ending cycle" (Grant 2006).

Table 2 provides an overview of the insurgency's adaptations in their five systems. These adaptations served to move the program to the right of its original position on the *x*-axis in Figure 6.

Table 2: Insurgent Adaptations

	Before	Adaptation	After
Goals	Different sub-goals, but general goal of evicting the US military	Recognition that hurting government and civilians caused greater problems for US	Causing instability in the new government (later, sectarian attacks)
Product or Service	Ineffective attacks	Improved tactics, tools, processes, and organization	Effective attacks
Process	Simple attacks; no set tactics, techniques, or procedures	The sharing of knowledge and coordination between groups on tool use, tactics, and procedures	Better reconnaissance, supply chain, tactics, techniques, and procedures; more structured
Organization	Coordination only within a cell	Increased their network, despite different micro goals; used more experts and middle-men	Coordination between cells; deference to shared goals; formal liaisons and information brokers mediating between cells
Tools	Ineffective small arms attacks	Technological advancements in IEDs	Increase in size and complexity of IEDs

3.4.2 Second Case Couplet: Microsoft and Google

(A) Microsoft's Longhorn Program

For a software company, Microsoft can be positioned on the right side of the *x*-axis in Figure 6. During its early years, Microsoft was extremely informal and on the cutting edge of the software industry. As it grew in size, its structures became more complex. The company grew a hierarchal structure where decisions were made at the top and filtered down. Most new product decisions must be approved from the top, usually by Bill Gates. This creates a long time lag that hinders innovative product development. Furthermore, Microsoft's many large divisions slow down innovation, as they are bogged down with the goal of defending the Windows operating system (OS) and Office Suite revenue streams. Haeckel (1999) describes a make-and-sell model, which Microsoft fits into: "It does what it already knows through forecasting, minimizing the cost and expense of making and selling its products. Leadership in a make-and-sell organization operates a closed system, ignoring for as long as possible signals that a change may be required." Microsoft's product roll-out times were lagging behind other companies', and its products were not as innovative. "Microsoft is not an agile competitor in the absolute sense – they simply have better strategic moves and more assets to leverage than anyone else who has stepped into the ring" (Dove

2001). These arrangements affected Microsoft's overall competitiveness, and Microsoft recognized the need to adapt.

The Longhorn program, which is developing the new Windows Vista OS, provides examples of Microsoft's changes. Microsoft's normal process for developing an OS is to have separate software engineers work on their own code and then submit it each night for a build (Guth 2005a). These builds helped to identify any bugs. But Longhorn's size and complexity was not conducive to this approach. The process of fixing the bugs would entail searching through thousands of lines of code.

To turn the Longhorn program around, Microsoft brought on several key personnel, including changing the head of the Windows group, which is the largest revenue and profit center. The *status quo* is for a software engineer to head this division, but Microsoft changed this paradigm by selecting an individual with a non-engineering background, but with excellent management skills (Guth 2006). This reorganization is an attempt to bring management control to the Windows group and hopefully get the Longhorn project back on track. The change is a result of recognizing a value gap with wide implications for Microsoft, including the obvious value gap caused by missing the roll-out date for the Windows Vista OS. Microsoft continues to refine its organizational structure in an effort to close the value gap.

In addition to changes in the organization system, the Longhorn program made changes in its other program systems. It completely revamped the software development process (Guth 2005b). It implemented a simpler product architecture where additional functionality could be added on later. It also better integrated their software engineers to boost coding efficiency (Guth 2005a). These changes took time to acknowledge and implement. Microsoft's program structures contributed to this delay.

Table 3 provides an overview of adaptations in the five systems of the Longhorn program.[5] Several of these adaptations served to move the program a bit to the left of its original position on the *x*-axis in Figure 6. The Longhorn case has several similarities with the Route Irish case. Both involve large enterprises needing to recognize a value gap between their current and impending outcomes and their

[5] We have not been able to dig very deep in this case and have relied so far on third-party sources of information, so the entries in Table 3 are unfortunately general at this point.

desired ones. Due to their size and the formality in their processes, both were somewhat slow to adapt to the current situation and competition. Two factors helped drive the changes: a clear competitor achieving visible successes and clear directions (goal formulations) from top leadership. Finally, both the US Army and Microsoft decreased the level of formality and structure in their systems to decrease their value gap.

Table 3: Microsoft Longhorn Program Adaptations

	Before	Adaptation	After
Goals	Defend the Windows OS and Office Suite	Changed goal to create the next-generation OS more efficiently	Create an innovative, robust new OS
Product or Service	Extremely complex OS product architecture	Simplified OS product architecture	Simpler OS product architecture
Process	Highly formal process with nightly builds and error tracing	Restructured software development process	More efficient and effective development process
Organization	Engineering-oriented program manager; less integrated software engineers writing code	Integrated software developers; brought in new program management and division leadership	Management-oriented program manager; improved integration of program organization
Tools	Using standard development tools	No major change discovered	Same

(B) Google's Innovation Continues

Google's organizational structure and goals are significantly different than Microsoft's. Google's goal is to catalog the world's information and make it accessible to people. They also have the underlying goal of creating innovative products. Google fits well into Haeckel's (1999) sense-and-respond model: "A sense-and-respond firm does not attempt to predict the future demand of its offerings. Instead, it identifies changing customer needs and new business challenges as they happen, responding to them quickly and appropriately." Google's process for developing software products consists of using small teams that brainstorm and develop the product until completion. These teams are then dismantled. The company's organizational structure is flat, networked, and based smart people on sharing ideas. This structure gives the company the ability to generate ideas that quickly become innovative products. Based

on their relatively unstructured systems for product development projects and programs, we place Google towards the left side of the *x*-axis in Figure 6.

Google is an interesting example of an enterprise that resists becoming more formal, despite a business environment that demands a more structured set of systems. Google's financing was provided by two venture capital firms. These two firms expected Google to follow the mold of traditional start-ups. Normally a traditional start-up would follow the advice and suggestions of its venture capitalists, but Google resisted pressures to move to a more formal organization structure, even when their own venture capitalists demanded it. The Google founders resisted hiring a CEO up to the last possible moment (Vise and Malseed 2005). When they eventually hired a CEO, they took the unusual approach of maintaining the majority of the power and control with the founders. Google became more structured in regards to building a corporate headquarters and creating a formal corporate staff. Wall Street brought additional pressures with the demand of knowing financial records and demanding to know the strategic direction Google was headed. The founders resisted these demands by maintaining secrecy when it came to revenue and future product developments (Vise and Malseed 2005). Despite these major changes, Google has largely retained its relatively informal processes for product development.

But Google saw the expediency of abdicating some of its freedom for the advantages of partnering structures. Google transformed from a start-up to a leading contender in the search engine industry through partnerships with Yahoo!, Ask Jeeves, and AOL. Google created credibility with Wall Street and enhanced its image by cooperating with high-visibility competitors. Where Microsoft was out to crush its competitors, Google was willing to work with and create profit-building partnerships with its competitors.

Table 4 provides an overview of Google's adaptations in their five systems.[6] These adaptations served to move it to the right of its original position on the *x*-axis in Figure 6. We can also draw similarities between Google and the insurgency. The insurgency grew in strength only after coordinating

[6] We have not been able to dig very deep in this case and have relied so far on third-party sources of information, so the entries in Table 4 are unfortunately general at this point.

and creating a larger network to achieve its goals. The same can be said of Google; it created partnerships and expanded its brand image to move towards its goal of becoming the best Internet search engine.

Table 4: Google's Program System Adaptations

	Before	Adaptation	After
Goals	Develop innovative products that change the way the world catalogs and accesses information; private company	Expanded vision of the implications of goal; going public brought additional resources but also additional requirements and constraints	Basically the same, but expanded, and venturing out to other endeavors; public company
Product or Service	Initial venture was a superior search engine	New products: e.g., Gmail, Google Talk, Earth, Desktop, Maps	Quickly developed products with possibilities for refinement
Process	Worked independently and very secretively about its processes and finances	No known internal development process changes, but formalized financial reporting processes	Continues to use fairly informal development processes; formal financial disclosure procedures
Organization	Small groups brainstorming ideas; run by founders; very independent	Hired CEO; partnered with other companies, including some competitors	Continues to use small groups for ideas and product development; in alliances with other companies
Tools	Massive computer/server infrastructure used for the search engine	Adapting to demand by increasing the size of the infrastructure; likely adopting new tools at the micro level	Larger and still growing infrastructure; micro tool set has likely evolved with the state of the art

3.5 Managerial Insights

A variety of insights from these cases can be applied to the business world. A critical one is the importance of quick recognition of a value gap and a desire to address it. The US Army and Microsoft illustrate that a large, complex enterprise can be slow to react to a threat, even when the threat is known. Conversely, smaller, more agile enterprises recognize the value gap quickly and adapt their processes to close the gap. What they give up in resources, more nimble programs compensate for with quick adaptation.

A program can address a value gap in two ways—by improving its outcomes or changing its

goals. It will require further study to determine what factors influence the most appropriate response, and if structure and formality are included in this list. Since all enterprises will tend to spin their goals differently according to what they think external constituencies want to hear—downplaying or highlighting supposed goal changes—it can be difficult to measure goal change exactly. External groups can exert pressure to "do something different" or "stay the course." While the Route Irish program and the insurgency both adapted their goals, it is less clear that Microsoft and Google have made any significant changes to their overarching goals, but this could be the result of the differing depth and unit of analysis in these two case study couplets.

A program's structure plays a role in how fast it adapts. A flat, network-oriented organization structure seems to adapt more quickly than a hierarchical one, as evinced by Google and the insurgents. Information flow is one of the main reasons for this adaptability. Programs with lateral information flow are able to share information at a faster rate, which gets important issues before decision makers more quickly and effectively. However, without clearly channel definition and management, a completely lateral organization can quickly get overwhelmed with information. A hierarchal organization usually must process critical information from the bottom up, and the results of decisions must flow back down to their effectors. A program's structure is a critical element to its facilitation of change and the rate of change.

Managers possess the ability to make their areas more adaptable. A manager may not be able to change an entire program, but he or she can start with the goals, product, process, organization, and tools in his or her area. First, a manager should flow down goals only after conducting a thorough analysis of the higher-level goals, the external environment, and the internal capabilities (people, processes, and tools). Second, a manager must honestly recognize any value gap. The time it takes to recognize a value gap is the reason a program must go through dramatic change as opposed to incremental change. Admitting there is a value gap is the biggest hurdle for a manager to overcome in a formal and highly structured program. Third, managers should identify high-leverage areas for internal change. For example, while a new tool or technology might offer the promise of "twice the speed" or "thrice the

performance," many such improvements merely amount to the situation of a formula one race car stuck in a rush hour traffic jam. The promises were based on test-track performance, while actual use must occur within the constraints of the program's people, processes, and other tools. To avoid such local optima in program improvement, managers need a model of the program architecture wherein various improvement options and scenarios can be pre-tested. Fourth, in addition to making specific improvements to the program's systems, a manager should create an internal culture of adaptability.

In many cases, a program manager is highly constrained in his or her ability to change people, processes, tools, and goals. A prime example is the Army's inability to procure the proper armored vehicles for use in Iraq. They developed a vehicle called the Joint IED Neutralizer, but they have not yet shipped it to Iraq (Mazzetti 2006). The delay is a result of a formal enterprise and its inherent bureaucracy. On the other hand, informal enterprises also have difficulty procuring the proper tools to support their stated goals. They do not have the same type of access to resources or the infrastructure to develop the right tools.

When considering potential changes to people, processes, and tools, a manager must ask critical questions about the impact the change will have on other systems, and on future adaptability. Does the change create more standardization, which might inhibit future changes? Does it help the program systems work together? A degree of risk is involved in making any changes. Depending on the enterprise, employees may be more likely to be punished for risky *action* than for risky *inaction* (Sitkin and Pablo 1992). This obstacle must be overcome to accelerate adaptation. One tool that a manager can use is a web-based forum for information-sharing. The transformation is derived by replacing the one-way flow of information of a program with fluid, on-line conversations (Dixon 2006). While this step does not instantly move a program to the left on the *x*-axis in Figure 6, it does set the conditions for further transformation. Making a too-formal program less formal can allow greater adaptability and enhance value.

How can these observations help managers and leaders in their respective industry? A manager's analysis into their own program's structure will provide some insight on how fast they can adapt their

systems to changes in the marketplace. They will also be able to discern the situation where a program recognizes a value gap but is slow to close it. These observations can lead to situation-appropriate actions. Recognition of these common trends is vital to their prevention. By avoiding these pitfalls, a program can increase their adaptability and the emergence of value.

4. Conclusion

This paper has presented some rough ideas on the subjects of program architecture, modeling it, adapting it, and getting the most value from it. Much further thinking and research is needed in these areas. My current research is looking at various sub-elements of the ideas in this paper, but I expect there are great possibilities in looking at them more holistically. I welcome comments and feedback on any of these ideas.

5. References

Alexander, Christopher (1964) *Notes on the Synthesis of Form*, Cambridge, MA: Harvard University Press.

Allen, Thomas J. (1977) *Managing the Flow of Technology: Technology Transfer and the Dissemination of Technological Information Within the R&D Organization*, 1984 Paperback Edition, Cambridge, MA: MIT Press.

Bartolomei, Jason E., Daniel E. Hastings, Richard de Neufville and Donna H. Rhodes (2006) "Screening for Real Options 'In' an Engineering System: A Step Towards Flexible System Development, PART I: The Use of Coupled Design Matrices to Create an End-to-End Representation of a Complex Socio-Technical System," *Proceedings of the 16th Annual International Symposium of INCOSE*, Orlando, FL, Jul 9-13.

Browning, Tyson R. (1999) "Designing System Development Projects for Organizational Integration," *Systems Engineering*, **2**(4): 217-225.

Browning, Tyson R. (2001) "Applying the Design Structure Matrix to System Decomposition and Integration Problems: A Review and New Directions," *IEEE Transactions on Engineering Management*, **48**(3): 292-306.

Browning, Tyson R. and Steven D. Eppinger (2002) "Modeling Impacts of Process Architecture on Cost and Schedule Risk in Product Development," *IEEE Transactions on Engineering Management*, **49**(4): 428-442.

Browning, Tyson R., Ernst Fricke and Herbert Negele (2006) "Key Concepts in Modeling Product Development Processes," *Systems Engineering*, **9**(2): 104-128.

Browning, Tyson R. and Ranga V. Ramasesh (2007) "A Survey of Activity Network-based Process Models for Managing Product Development Projects," *Production and Operations Management*, **16**(forthcoming).

Chiarelli, P.W. (2006) V Corps Assumption of Command Speech. Baghdad, Iraq.

Danilovic, Mike and Tyson R. Browning (2007) "Managing Complex Product Development Projects with Design Structure Matrices and Domain Mapping Matrices," *International Journal of Project Management* (forthcoming).

Dixon, Nancy M (2006) "Peer-to-Peer Leadership Development," *Harvard Business Review*: 56-57.

Dove, Rick (2001) *Response Ability The Language, Structure, and Culture of the Agile Enterprise*, New York: John Wiley & Sons, Inc..

Else, Steven E. (2004) *Organization Theory and the Transformation of Large, Complex Organizations: Donald H. Rumsfeld and the U.S. Department of Defense, 2001-04*, Thesis (The Faculty of the Graduate School of International Studies), University of Denver, Denver.

Eppinger, Steven D. and Vesa Salminen (2001) "Patterns of Product Development Interactions," *Proceedings of the International Conference on Engineering Design (ICED)*, Glasgow, Aug 21-23.

Feldman, Martha S. (2000) "Organizational Routines as a Source of Continuous Change," *Organization Science*, **11**(6): 611-629.

Fixson, Sebastian K. (2005) "Product Architecture Assessment: A Tool to Link Product, Process, and Supply Chain Design Decisions," *Journal of Operations Management*, **23**: 345-369.

Galbraith, Jay R. (1994) *Competing with Flexible Lateral Organizations*, Second Edition, Reading, MA: Addison-Wesley.

Galbraith, Jay R., Edward E. Lawler III and Associates (1993) *Organizing for the Future: The New Logic for Managing Complex Organizations*, San Francisco: Jossey-Bass.

Ghosh, A. (2005) "Professor Of Death," *Time*.
Grant, Greg (2006) "Coalition Faces More Frequent, Sophisticated Military Operations," *Defense News*, **21**(5): 22.
Guth, Robert A. (2005a) Battling Google, Microsoft Changes How It Builds Software, *Wall Street Journal*. New York, NY.
Guth, Robert A. (2005b) Microsoft Sets Big Restructuring Plan, *Wall Street Journal*. New York, NY.
Guth, Robert A. (2006) As Microsoft Fumbles, Marketers Supplant Tech Managers, *Wall Street Journal*. New York, NY.
Haeckel, Stephan H. (1999) *Adaptive Enterprise: Creating and Leading Sense-and-Respond Organizations*, Boston, MA: Harvard Business School Press.
Holland, John H. (1998) *Emergence: From Chaos to Order*, Reading, MA: Helix (Addison-Wesley).
Hutchins, Edwin (1991) "Organizing Work by Adaptation," *Organization Science*, **2**(1): 14-39.
Jervis, R. (2005) Pressure-Triggered Bombs Worry U.S. Forces, *USA Today*.
Kotter, John (1990) *Force for Change: How Leadership Differs from Management*, New York, NY.: The Free Press.
Levitt, Barbara and James G. March (1988) "Organizational Learning," *Annual Review of Sociology*, **14**: 319-340.
Lorsch, Jay W. (1982) "Organization Design: A Situational Perspective," in Tushman, Michael L. and William L. Moore, Ed., *Readings in the Management of Innovation*, Cambridge, MA: Ballinger pp. 477-488.
Lorsch, Jay W. and Paul R. Lawrence, Eds. (1972) *Managing Group and Intergroup Relations*, Homewood, Ill.: Richard D. Irwin.
Mazzetti, Mark (2006) Bomb Buster for Iraq Hits Pentagon Snag, *Los Angeles Times*. Los Angeles.
Mintzberg, Henry (1979) *The Structuring of Organizations*, Englewood Cliffs, NJ: Prentice-Hall.
Nagel, J. A. (2002) *Learning to Eat Soup with a Knife*, Chicago: The University of Chicago Press.
Negele, Herbert, Ernst Fricke and Eduard Igenbergs (1997) "ZOPH--A Systemic Approach to the Modeling of Product Development Systems," *Proceedings of the 7th Annual International Symposium of INCOSE*, Los Angeles, Aug. 3-7, pp. 773-780.
PMI (2004) *A Guide to the Project Management Body of Knowledge*, 3rd Edition, Newtown Square, PA: Project Management Institute.
Rechtin, Eberhardt (1991) *Systems Architecting: Creating & Building Complex Systems*, Englewood Cliffs, NJ: PTR Prentice Hall.
Sanchez, Ron and Joseph T. Mahoney (1996) "Modularity, Flexibility, and Knowledge Management in Product and Organization Design," *Strategic Management Journal*, **17**(Winter): 63-76.
Sharman, David M. and Ali A. Yassine (2004) "Characterizing Complex Product Architectures," *Systems Engineering*, **7**(1): 35-60.
Simon, H.A. (1962) "The Architecture of Complexity," *Proceedings of the American Philosophical Society*, **106**: 467-482.
Sitkin, Sim B. and Amy L. Pablo (1992) "Reconceptualizing the Determinants of Risk Behavior," *The Academy of Management Review*, **17**(1): 9-38.
Sosa, Manuel E., *et al.* (2002) "Factors That Influence Technical Communication in Distributed Product Development: An Empirical Study in the Telecommunications Industry," *IEEE Transactions on Engineering Management*, **49**(1): 45-58.
Sosa, Manuel E., Steven D. Eppinger and Craig M. Rowles (2003) "Identifying Modular and Integrative Systems and Their Impact on Design Team Interactions," *Journal of Mechanical Design*, **125**(2): 240-252.
Spear, Steven and H.K. Bowen (1999) "Decoding the DNA of the Toyota Production System," *Harvard Business Review*: 96-106.
Staudenmayer, Nancy, M. Tyre and L. Perlow (2002) "Time to Change: Temporal Shifts as Enablars of Organziational Change," *Organization Science*, **13**(5): 583-597.
Strandberg, Tom, Herb Burton and Dinesh Verma (2006) "On the Alignment between System Architectures and Organizational Structures," *Proceedings of the 16th Annual International Symposium of INCOSE*, Orlando, FL, Jul 9-13.
Thompson, James D. (1967) *Organizations in Action*, New York: McGraw-Hill.
Tushman, Michael L. and David A. Nadler (1978) "Information Processing as an Integrating Concept in Organizational Design," *Academy of Management Review*, **3**(3): 613-624.
Ulrich, Karl T. (1995) "The Role of Product Architecture in the Manufacturing Firm," *Research Policy*, **24**(3): 419-440.
Vise, D. and M. Malseed (2005) *The Google Story*, New York, NY: Bantam Dell.