Framework for the Reduction of Programmatic Risk on Complex Systems Projects

Mark Minnucci

B.S. Electrical Engineering
Villanova University, 2003

Submitted to the System Design and Management Program
in Partial Fulfillment of the Requirements for the Degree of:

Master of Science in Engineering and Management

At the
Massachusetts Institute of Technology
January 2010
[February 2010]

Author: _____
Mark Minnucci
System Design and Management Program
January 15, 2010

Certified by: _____
Dr. Nancy Leveson
Thesis Supervisor
Department of Aeronautics and Astronautics

Certified by: _____
Patrick Hale
Program Director
System Design and Management Program

Abstract

Framework for the Reduction of Programmatic Risk on Complex Systems Projects

by Mark Minnucci

Submitted to the System Design and Management Program on
January 15, 2009 in Partial Fulfillment of the Requirements for the
Degree of Master of Science in Engineering and Management

"In 2008, the cumulative cost growth in the Department of Defense's (DoD) portfolio of 96 major defense acquisition programs was $296 billion and the average delay in delivering promised capabilities to the warfighter was 22 months." This statement from the Director of Acquisition and Sourcing Management of the Government Accountability Office (GAO) before a House of Representatives panel is in reference to an alarming, decades-long trend in the Defense Industry of budget and schedule overruns. Defense projects are complex systems of humans, software, and hardware interacting in unpredictable and often-uncontrolled ways. The research presented in this thesis demonstrates that component and systemic failures in DoD systems have much in common with the overruns that their executing organizations experience.

Complex systems accidents occur when their control mechanisms do not sufficiently enforce constraints on system components and their interactions. Similarly, project losses, in terms of budget and schedule overruns, occur when the control mechanisms of the executing organization do not sufficiently enforce constraints on project teams and their interactions. This thesis proposes a framework based on the principles of Control Theory, Systems Safety Analysis, and Earned Value Management, which project managers can apply in order to reduce programmatic risk on complex systems projects.

The objectives of the thesis are: to provide project managers with a mechanism to control risk within the scope of the work they oversee, to provide individual contributors with a mechanism to control risk within the scope of the work they execute, to clearly demonstrate how poorly designed organization structures facilitate program losses, and to clearly demonstrate how well-designed organization structures can prevent or at the minimum mitigate program losses.

At the completion of this thesis, it was found that complex systems programs have many tools at their disposal for defining relationships between elements of project scope and between teams in the executing organization. But few tools are available to specify how exactly a manager can accurately monitor and safely affect the scope elements under their control. The control structure specification and design presented within this thesis will address the primary causes of risk that lead to program losses.

Thesis Supervisor: Dr. Nancy Leveson
Title: Professor of Aeronautics, Astronautics and Engineering Systems

Acknowledgements

This thesis is the culmination of a three-year journey. In 2006, I decided to pursue a graduate degree that would both grow my systems engineering expertise and simultaneously provide me with a foundation in management theory. There are only a few such programs in the country, and I had the good fortune of living only thirty minutes away from the best of these programs. The Massachusetts Institute of Technology (MIT) System Design and Management (SDM) program was exactly the education for which I was looking. The time spent interacting with MIT faculty, completing SDM coursework, collaborating with members of my cohort, and developing this thesis was without question the most enlightening of my educational career. However, without the support of a few key individuals none of this would have been possible. I would like to take this brief opportunity to express my gratitude.

The very first person I met at MIT was Pat Hale, Director of the SDM program. Pat gave me a brief tour of the MIT campus, allowed me to sit in on a SDM class, and entertained all of my questions. With his approval, I was accepted into the SDM program and began my coursework in the fall of 2006. Over the past three years, Pat has always kept his door open and made time to advise me on my graduate work. He has been incredibly flexible in tailoring my degree to fit both my personal interests and my full-time work schedule. Thank you Pat, for letting me be a part of SDM and for all you have done for me over these past three years.

The first professor I had at MIT was Dr. Nancy Leveson, professor of Aeronautics, Astronautics and Engineering Systems. Through her Software Engineering Concepts course, I was prepared for the critical thinking skills that would be demanded of me throughout my time at MIT. Nancy's lectures were not only informative but were also supplemented by a deep work history in the Department of Defense to which I could relate. Thank you Nancy, your course was the first I took at MIT, and it has been very rewarding to have you as the final influence during the development of this thesis.

My time at MIT would not have been possible were it not for the extreme generosity of my employer, Raytheon Company. In 2007, I was honored to receive an Advanced Study Program scholarship covering my books and tuition. There were many individuals who were advocates for me during the scholarship process. Thank you Dan Dechant, Tim Scheve, Caroline Elias, Paul Bailey, and Laura Dauphinais. I am also very grateful to the managers who helped me maintain a balance between work and school over the past three years. Thank you Carly Halloran and Carrie Schoenholtz.

Last, I would like to thank my wife Bridget for all she has done for me while I was working and in school. You have been patient and understanding of my schedule and always willing to listen when the workload piled up. More importantly, without your encouragement I would never have applied to MIT or to the Advanced Study Program. Thank you, you have earned this degree as much as I have.

Table of Contents

List of Figures

List of Tables

1 Introduction

"In 2008, the cumulative cost growth in the Department of Defense's portfolio of 96 major defense acquisition programs was $296 billion and the average delay in delivering promised capabilities to the warfighter was 22 months [7, pg 1]." This was the opening statement from the Director of Acquisition and Sourcing Management of the Government Accountability Office (GAO) before a House of Representatives panel on April 1, 2009. The 96 major programs that the Director referred to had experienced a combined 25% cumulative cost overrun from their initial $1.28 trillion combined budgets. Unfortunately these statistics were both bad and consistent with data presented five years previous. In 2003, the GAO testified that the Department of Defense's (DOD) portfolio of 77 major defense programs had experienced a combined 19% cumulative cost overrun from their initial $1 trillion combined budgets and an average schedule delay of 18 months.

During his Congressional testimony, the Director outlined an approach to counteract the alarming trend of significant DOD program budget and schedule overruns [7]. He proposed that drastic improvements in the acquisition process could be realized if the GAO began monitoring three categories of data on major defense programs: *knowledge metrics*, *outcome metrics,* and *prerequisite indicators*.

- *Knowledge metrics* are a set of thresholds for technology maturity, design stability, and manufacturing maturity, to be examined at key program checkpoints. These checkpoints are respectively: the completion of system design, the middle of product development, and prior to the start of production.
- *Outcome metrics* are a set of high-level health indicators, taken annually, to monitor how well programs are being executed in terms of original baseline budget and schedule. The metrics are: development cost, procurement cost, total program cost, quantities to be procured, procurement unit costs, total program unit costs, and cycle time from Product Design Review to Initial Operational Capability.
- *Prerequisite indicators* are a set of best practices that can verify the realism of program acquisition plans. These best practices are: a viable business case, a distinct separation of research and production efforts, a realistic production schedule, an early systems engineering risk assessment, and a congressional commitment to fully funding approved projects.

The Director closed his testimony by emphasizing, "Critical to achieving successful outcomes is establishing knowledge-based, realistic program baselines. Without realistic baselines, there is no foundation for accurately measuring the knowledge and health of programs [7, pg 12]."

While the Director's statement is absolutely correct and while his approach may improve the acquisition process, it is the author's opinion that in five years, another GAO Director will once again be testifying in front of Congress about significant DOD budget and schedule overruns. The prevention and mitigation of these overruns, on DOD and other complex systems projects, is the central purpose of this thesis.

1.1 Thesis Statement

Complex systems accidents occur when their control mechanisms do not sufficiently enforce constraints on system components and their interactions [4, pg 4]. Similarly, program losses, in terms of budget and schedule overruns, occur when the control mechanisms of an organization do not sufficiently enforce constraints on project scope and the teams that execute that scope. This thesis proposes a framework based on the principles of Control Theory, Systems Safety Analysis, and Earned Value Management, which project managers can apply to both new and existing projects, in order to reduce programmatic risk on complex systems projects.

1.2 Objectives

To suggest that the objective of this thesis is to cure all of the budgetary issues on complex systems projects would be overly ambitious. Projects have, are, and always will run into some issues that will force a program to experience losses. Instead, it is the overall objective of this thesis to clearly define a control structure that project managers can embed within their organizations in the interest of reporting losses to appropriate controllers as soon as they occur, and preventing losses from occurring in the first place if at all possible. Along with this new organizational framework are a set of operating practices that will make project safety a part of the organization's culture. Together, the proposed framework and operating practices will reduce the risk on the smallest elements of project scope, which will result in lower risk in successively larger scope elements when aggregated higher and higher in the project hierarchy.

The specific objectives of this thesis are as follows:

- To provide project managers with a mechanism to control risk within the scope of the work they oversee
- To provide individual contributors with a mechanism to control risk within the scope of the work they execute
- To clearly demonstrate how flaws in an organizational control structure can facilitate program losses
- To clearly demonstrate how a well designed organizational control structure can mitigate or even prevent program losses

1.3 Motivation and Approach

In developing this thesis, the author drew inspiration from three primary sources. First, years of experience as both an engineer and engineering manager on several large DOD projects led the author to identify both the domain and need for the thesis. Second, coursework within the MIT System Design and Management program such as "System and Project Management," "System Architecture," and "Software Engineering Concepts" provided the author with an education in the core concepts on which the thesis itself was based. Lastly, the work of Dr. Nancy Leveson, on embedding safety into systems and the culture of the organizations that build them, provided the author with the focused inspiration to address the need for the thesis.

The approach for this thesis is as follows:

- Define a clear need for change in the existing organizations and processes used to manage complex systems projects
- Research the core concepts involved within the current methods of planning and executing complex systems projects: project management, organizational structure, and control systems
- Define a framework that synthesizes the core concepts into a representation of an organization as a hierarchical controller with the project scope as the controlled process
- Prove the validity of the proposed framework and operating policies in a realistic case study
- Apply safety analysis to illustrate the failings of the current methods of planning and executing complex systems projects and provide a set of recommendations for project managers on how to embed safety into their planning activities and operating policies

1.4 Document Organization

This document has been organized to present the thesis to the reader in a structured fashion. First, Sections 2 through 4 will introduce the reader to the key concepts necessary to develop the thesis. Second, Section 5 will present the central thesis. Finally, Section 6 will show an application of the thesis in a detailed case study.

The following are the major sections of this thesis:

- Section 2, "Projects and Project Management," introduces concepts in project definition such as scope, schedule and budget planning. The section also covers a complex systems project management methodology known as Earned Value Management.
- Section 3, "Organizations," introduces the concept of an organization structure and discusses the merits of several structures from which a project manager can choose. Later in the section, the relationships among organization structures, project scope, and funding sources are explored.
- Section 4, "Control Systems," introduces concepts in Control Theory such as: controllers, actuators, sensors, and controlled processes. The section then goes on to define common control system design patterns and the sources of errors that can be introduced into a controlled process.
- Section 5, "Integrated Framework," synthesizes the fundamental concepts presented in the previous three sections into a representation of an organization as a hierarchical controller with the project scope as the controlled process. The section shows how project teams and Earned Value Management metrics are direct equivalents to the control systems concepts of actuators and monitored variables. This section develops a specification for a structural framework to control programmatic risk and losses. The section closes with a proposed design for a control system that meets the developed specification.

- Section 6, "Case Study," applies the proposed thesis framework to an actual complex system project. The section will show the current state of the project and the structure of its executing organization. The section will then clearly identify areas where inadequate enforcement of budget and schedule constraints on elements of the project scope could lead to losses. This section closes by providing a set of recommendations for improving the state of the example project.
- Section 7, "Conclusion," summarizes the concepts presented in the thesis. Areas for future research are presented, as well as a final discussion of the benefits of applying the ideas proposed in the thesis.

2 Projects and Project Management

This section is an introduction to the concepts of project management. Thousands of books and articles have been published on every aspect of project management, and as such this section is only intended to cover the fundamentals necessary for understanding this thesis.

2.1 Defining Projects

A project is a series of interrelated tasks performed to meet one or more explicit goals, referred to as the project *scope*. Projects are executed through the application of resources, usually within a set of constraints. These constraints become the implicit goals of the project. The two most common constraints for a project are *schedule* and *budget*. Let us look at each of these three terms, *scope, schedule, and budget,* in detail and begin building an example project.

2.1.1 Scope

Scope is the central goal of any project. It is typically defined, at the highest level, in terms of one or more primary deliverables to a customer. So for instance, the scope of a software project may be defined as the delivery of executable code. On most projects, scope is sub-divided into smaller components; detail increasing with decomposition. One useful tool for organizing a project's scope is called a *work breakdown structure* (WBS).

A WBS is a hierarchical tree-structure that is organized around the primary deliverables of the project. It is critical that the highest level of the WBS contains exactly 100% of all work to be performed. Otherwise, gaps or overlaps in effort can occur. Each deliverable is assigned a unique ID number, then decomposed into tasks of lower, and roughly equivalent level of detail. These tasks are also assigned unique ID numbers and like the parent level, must total to 100% of all work to be performed. This process continues until the defined components of work are of sufficiently manageable size for teams or individuals to execute. Table 2.1-1 is an example of a very basic WBS.

Table 2.1-1 Sample Work Breakdown Structure

Deliverable	Task	Sub-Task	WBS ID
1. Deliver product A	1. Develop system design	1. Sub-Task 1	1.1.1
		2. Sub-Task 2	1.1.2
	2. Develop system requirements	1. Sub-Task 1	1.2.1
		2. Sub-Task 2	1.2.2
	3. Develop software requirements	1. Sub-Task 1	1.3.1
		2. Sub-Task 2	1.3.2
2. Deliver product B	1. Conduct trade study	1. Sub-Task 1	2.1.1
		2. Sub-Task 2	2.1.2
	2. Generate Report	1. Sub-Task 1	2.2.1
3. Deliver product C	1. Interview potential suppliers	1. Sub-Task 1	3.1.1
		2. Sub-Task 2	3.1.2
	2. Contract selected supplier	1. Sub-Task 1	3.2.1

A WBS does not need to be organized by customer deliverables at its highest level. A valid WBS could be constructed by other schemes such as phases of work (Design, Development, Testing, etc), or physical sub-systems (Engine, Chassis, Fuel, etc). The important point is that whatever organizational scheme is used, the structure should focus on desired goals of the project and remain relatively stable throughout the project lifespan.

Once the scope has been defined and sufficiently decomposed, dependencies between the components at each level should be captured. As an example, Table 2.1-2 below shows the relationships between components at the third level of scope.

Table 2.1-2 Sub-Task Dependencies

Project ID	Predecessor	Successor
1.1.1	START	1.1.2
1.1.2	1.1.1	1.2.1
1.2.1	1.1.2	1.2.2
1.2.2	1.2.1	1.3.1
1.3.1	1.2.2	1.3.2
1.3.2	1.3.1	FINISH
2.1.1	START	2.1.2
2.1.2	2.1.1	2.2.1
2.2.1	2.1.2	3.1.1
3.1.1	2.2.1	3.1.2
3.1.2	3.1.1	3.2.1
3.2.1	3.1.2	FINISH

Predecessor and successor relationships are gates that effectively bound tasks and communicate handoffs between teams. It is common on smaller projects to generate a dependency diagram, such as Figure 2.1-1, to aid in the understanding and analysis of the scope and its component relationships.

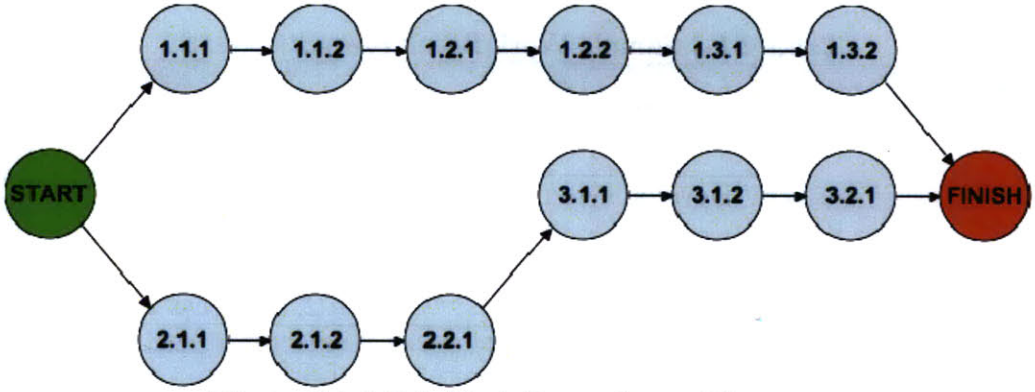

Figure 2.1-1 Sub-Task Dependency Diagram

Now that the scope has been characterized for the example project, let us move on to *schedule*, the first major project constraint.

2.1.2 Schedule

Schedule is the extent of time available to complete a project. It is typically bounded, at the highest level, by calendar start and end dates of relevance to the customer. So for instance, the schedule of a construction project may start on the date of contract award, and end on the date of final building inspection. On most projects, the overall schedule is sub-divided and allocated to the tasks that comprise the project scope. Table 2.1-3 below, is a hypothetical schedule for the example project.

Table 2.1-3 Sample Schedule

Project ID	Plan Start	Plan Finish
1	**01/01/09**	**12/31/09**
1.1.1	01/01/09	02/28/09
1.1.2	03/01/09	04/30/09
1.2.1	05/01/09	06/30/09
1.2.2	07/01/09	08/31/09
1.3.1	09/01/09	10/31/09
1.3.2	11/01/09	12/31/09
2	**01/01/09**	**06/30/09**
2.1.1	01/01/09	02/28/09
2.1.2	03/01/09	04/30/09
2.2.1	05/01/09	06/30/09
3	**04/01/09**	**12/31/09**
3.1.1	07/01/09	08/31/09
3.1.2	09/01/09	10/31/09
3.2.1	11/01/09	12/31/09

Depending on the nature and granularity of the scope, it may be beneficial to provide the number of working days associated with each of the planned start and finish dates. Looking at sub-task 1.1.1 for instance, the planned start and finish would appear to indicate that 59 days of schedule were available. For business purposes however, this time period contains only 42 working days once weekends are discounted.

An important point to note about schedule is that it should not be confused with *effort*. Effort is a measure of the amount of resources required to complete a task. Schedule is the period of calendar time available to apply those resources. So again looking at sub-task 1.1.1, while 42 working days of schedule are allocated to the task, it may only require 30 person-days of effort. This distinction between schedule and effort will be explored as part of the second major project constraint, which is budget.

2.1.3 Budget

Budget is the amount of financial resources available to execute a project. It is typically bounded, at the highest level, by a maximum amount that the customer is willing to spend to complete the project's scope. Once an overall budget has been assigned to a project, project managers partition the budget into accounts such as labor, materials, travel, emergency reserve, etc. For the example project, we will only consider the labor portion

of the budget from this point forward. Just as the schedule was sub-divided and allocated to the tasks that comprise the project scope, so to is the budget. Table 2.1-4 below, is a hypothetical budget for the example project.

Table 2.1-4 Sample Budget

Project ID	Plan Budget ($)	Plan Budget (Person-Hrs)
1	**500,000**	**5000**
1.1.1	100,000	1000
1.1.2	100,000	1000
1.2.1	100,000	1000
1.2.2	100,000	1000
1.3.1	50,000	500
1.3.2	50,000	500
2	**300,000**	**3000**
2.1.1	190,000	1900
2.1.2	90,000	900
2.2.1	20,000	200
3	**200,000**	**2000**
3.1.1	120,000	1200
3.1.2	30,000	300
3.2.1	50,000	500

It is obvious why a project manager would want to express budget in dollars, but it may not be so obvious why expressing budget in hours can also be very useful. There are three reasons. First, during both bidding and execution it is easier conceptually to understand spending labor in terms of hours as opposed to spending labor in terms of dollars. Most engineers can define technical work in hours, but are at a loss when it comes to the cost of those hours. This can be attributed to the fact that the hourly salary for an engineer is only a portion of what goes into the hourly labor rate billed to the customer for one hour of an engineer's time. Other aspects of the hourly labor rate include overhead, profit, management time, etc. The second reason, related to the first, is that the hourly labor rate can change over the course of the project schedule. So on some contracts, the customer may request to see the budget in terms of hours, which will hopefully remain more stable than the dollar budget. The third and final reason to express budget in terms of hours is that it allows the project manager to make a connection between budget and schedule.

This last point ties back to the discussion on the difference between schedule and effort. Sub-task 1.1.1 has 42 working days of schedule, and now 1000 person-hours of budget, allocated towards its completion. In terms of the calendar, 42 days translates to 1008 calendar hours. So, if a single engineer could work non-stop every one of those 42 days, they could conceivably finish 1.1.1, on budget, and 8 hours ahead of schedule. However, this is clearly an impractical plan. It would be much more reasonable to apply more employees to the completion of this work, which would result in multiple hours of budget (i.e. effort) being spent during each hour of schedule. Having a budget expressed in

person-hours reveals to the project manager new information about the level of staffing needed to execute the project that dollars alone would not have made apparent.

At this time, it should be clear that scope, schedule, and budget are not independent goals of the project. In fact, these three concepts are so interconnected, that they are commonly referred to as the "Project Management Triangle."

2.1.4 The Triangle

The so-called "Project Management Triangle" is a qualitative representation of the relationships between a project's scope, schedule, and budget. It is depicted as an equilateral triangle, such as Figure 2.1-2 below, with each goal forming one side of the triangle. The Triangle is meant to remind project managers that an increase or decrease in any one of the goals could have a positive or negative impact on the other two goals.

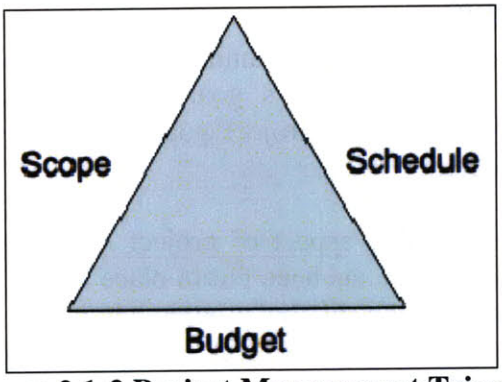

Figure 2.1-2 Project Management Triangle

While the Triangle is an overly simplistic depiction of project management relationships, it does introduce two new concepts: changes in magnitude and changes in rate. As an example, let us say that the project is in progress and the project manager realizes that a significant segment of the scope was overlooked. Adding this new work to the original scope translates literally to an increase in the length of the scope side of the Triangle. The manager could proportionally increase the magnitude of both the schedule and budget sides, and maintain the original angles of the Triangle. In many cases however, this kind of increase is not acceptable to customers. Instead, the manager may be forced to maintain the original budget and/or schedule. Based on the Triangle, this inherently means that the scope-budget, scope-schedule and/or budget-schedule angles must be adjusted. Without going through every permutation, one effect of scope increases, with fixed budget or schedule could be an increase in the rate of budget spent per schedule elapsed. In practical terms, this translates to either hiring more staff to execute more work in the same amount of time, or keeping the same staff and having them work more hours in the same amount of time.

These kind of qualitative illustrations are really the limit of the Triangle's usefulness. One shortcoming of the Triangle is that it implies that there are only three interdependent goals on the project. The Triangle rapidly becomes a spider web when a manager attempts to include equally critical constraints such as quality, staffing, and safety. A

second shortcoming of the Triangle is that the magnitude of all three sides could remain fixed and yet vastly different project outcomes could occur. A perfect example of this phenomenon appears when you consider staff efficiency. A project may have sufficient schedule and budget for the proposed scope if experienced employees are working it. However with inexperienced employees, significantly more schedule and budget may be required. A final shortcoming of the Triangle is that it does not provide a quantitative means for the project manager to analyze the project. Merely knowing that the schedule will be impacted by scope changes gives the manager no measure as to the size or duration of the impact.

We now have a plan in place for our example project. With scope, schedule, and budget all defined, and a basic understanding of their relationships to each other, let us now move on to the subject of how to manage projects.

2.2 Managing Projects

Project management is the discipline of planning and managing resources to bring about the successful completion of a project's goals. The primary challenge of project management is to achieve all of the project goals and objectives while honoring the preconceived project constraints.

The fundamentals of the planning aspect of project management were covered in the previous section. But once a plan has been put in place and execution begins, how can a project manager successfully monitor the progress of every task, identify issues in a timely fashion, and communicate status to team members and customers in a meaningful way? One widely used method to address each of these needs is through Earned Value Management.

Earned Value Management (EVM) is a quantitative methodology for tracking and predicting the progress of a project [2]. It is a collection of management practices that is scalable to projects of all sizes, although its greatest value is realized when applied to large-scale efforts. EVM was originally developed for use on Department of Defense (DoD) projects, but is now widely used in the private sector as well. The following section is a brief introduction to the key concepts of EVM.

2.2.1 The Value Plan

Earned Value Management, as the name may imply, is centered on the concept of *value*. Value in this context refers to effort expended towards accomplishing the project scope. A unit of measure is needed in order to quantify value for each of the sub-tasks defined during the planning phase. It is useful at this point to once again consider sub-task 1.1.1 of our example project. Sub-task 1.1.1 has 1000 person-hours of budget assigned. It stands to reason that at the start of the task if we have spent 0 person-hours on 1.1.1, then we have accomplished 0% of the sub-task's value. Likewise, assuming we have allocated funding perfectly, once all 1000 person-hours of budget has been spent, 100% of the sub-task's value should have been accomplished. Therefore, 1000 man-hours is the *Planned Value* of sub-task 1.1.1.

Planned Value (PV), sometimes referred to as the Budgeted Cost of Work Scheduled (BCWS), is the number of units of accomplishment we expect a task to take to complete. PV is tied back to the project schedule by assigning these units of accomplishment to calendar days. The sum of all PV on a project is referred to as the project's Budget at Completion (BAC). Table 2.2-1 shows a possible Planned Value schedule for our example project.

Table 2.2-1 Planned Value Schedule

ID	PV	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
1.1.1	1000	500	500										
1.1.2	1000			500	500								
1.2.1	1000					500	500						
1.2.2	1000							500	500				
1.3.1	500									250	250		
1.3.2	500											250	250
2.1.1	1900	950	950										
2.1.2	900			450	450								
2.2.1	200					100	100						
3.1.1	1200							600	600				
3.1.2	300									150	150		
3.2.1	500											250	250
	Monthly	**1450**	**1450**	**950**	**950**	**600**	**600**	**1100**	**1100**	**400**	**400**	**500**	**500**
	Cum	**1450**	**2900**	**3850**	**4800**	**5400**	**6000**	**7100**	**8200**	**8600**	**9000**	**9500**	**10000**

A few things are worth noting about the above table. First, the PV for each sub-task has been distributed evenly across each month of schedule for that sub-task. In reality, the PV should be tied to the schedule by clearly defined milestones with a portion of the sub-task's total PV assigned to each milestone. Second, the PV for all sub-tasks in a given month is shown at the bottom of the table. This is useful for comparing expected work from month to month. Third, the cumulative PV for all sub-tasks is also shown at the bottom of the table. This is useful for tracking progress of the project as a whole. Here you will note that the cumulative PV for the example project is 10,000 hours. This is the BAC for the example project.

2.2.2 Earning Value and Monitoring the Schedule

Earned Value (EV), sometimes referred to as the Budgeted Cost of Work Performed (BCWP), is the number of units of accomplishment completed towards the PV of a given task. At regular intervals, a project manager will claim credit for accumulated EV on each task of the project. A task is completed when the task's EV equals its PV. The project is completed when the sum of the entire project's EV equals the BAC. Table 2.2-2 below, shows how a project manager might capture the PV and EV status of the example project's Deliverable 1 at the end of March 2009.

Table 2.2-2 Deliverable 1 March 2009 PV and EV

	2009													
	Jan	**Feb**	**Mar**	**Apr**	**May**	**Jun**	**Jul**	**Aug**	**Sep**	**Oct**	**Nov**	**Dec**	**Current Month**	**Cum**
PV	500	500	500	500	500	500	500	500	250	250	250	250	500	1500
EV	400	400	800	0	0	0	0	0	0	0	0	0	800	1600

From the above table, we can learn a few things about Deliverable 1. First, the Current Month column tells us that the effort accomplished in March was well ahead of the planned effort for March (800 earned vs. 500 planned). Second, the Cumulative column tells us that the overall effort accomplished to date is slightly ahead of the planned effort for the same time period (1600 earned vs. 1500 planned). But EVM can provide much better quantitative analysis than "well ahead" and "slightly ahead." The following are three simple formulas that EVM project managers use to monitor their schedules:

- Schedule Variance (SV)
  - The number of hours the effort is ahead or behind the plan. A positive value is ahead of schedule, a negative value is behind schedule, and zero is on schedule.
  - $SV = EV - PV$
- Schedule Variance Percentage (SV%)
  - The percentage of hours the effort is ahead or behind the plan. A positive percentage is ahead of schedule, a negative percentage is behind schedule, and zero percent is on-schedule.
  - $SV\% = \frac{SV}{PV} \times 100$
- Schedule Performance Index (SPI)
  - An efficiency factor comparing progress relative to the plan. A value greater than one is ahead of schedule, a value less than one is behind schedule, and a value of one is on-schedule.
  - $SPI = \frac{EV}{PV}$

So with respect to schedule, Deliverable 1's status is as follows:

- Current month: SV = 300 hours, SV% = 60%, SPI = 1.6. The deliverable is 300 hours (i.e. 60%) ahead of the plan for the current month. The team accomplished 1.6 units of actual work for every 1 unit of planned work for the current month.
- Cumulative: SV = 100 hours, SV% = 7%, SPI = 1.07. The deliverable is 100 hours (i.e. 7%) ahead of plan for the project to date. The team accomplished 1.07 units of actual work for every 1 unit of planned work on average.

2.2.3 The Cost of Value and Monitoring the Budget

Actual Cost (AC), sometimes referred to as the Actual Cost of Work Performed (ACWP), is the number of units of cost spent accomplishing the EV of a given task. At regular intervals, a project manager will record the labor charges (hours in our example case) against each task of the project. When completed, a perfectly budgeted task's AC will

equal that task's PV. Likewise, on a perfectly budgeted project, the sum of all tasks' AC will equal the BAC. Table 2.2-3 below, shows how a project manager might capture the AC status of the example project's Deliverable 1 at the end of March 2009.

Table 2.2-3 Deliverable 1 March 2009 AC

	Deliverable 1 - 2009													
	Jan	**Feb**	**Mar**	**Apr**	**May**	**Jun**	**Jul**	**Aug**	**Sep**	**Oct**	**Nov**	**Dec**	**Current Month**	**Cum**
PV	500	500	500	500	500	500	500	500	250	250	250	250	500	1500
EV	400	400	800	0	0	0	0	0	0	0	0	0	800	1600
AC	400	400	900	0	0	0	0	0	0	0	0	0	900	1700

From the above table, we can learn a few new things about Deliverable 1. First, the Current Month column tells us that the budget expended in March was slightly ahead of the value earned for March (900 spent vs. 800 earned) and well over the planned value for March (900 spent vs. 500 planned). Second, the Cumulative column tells us that the overall budget expended to date is slightly ahead of the value earned to date (1700 spent vs. 1600 earned) and similarly over the planned value for the same time period (1700 spent vs. 1500 planned). The following are three simple formulas that EVM project managers use to monitor their costs:

- Cost Variance (CV)
  - The number of labor hours the effort is under or over the value earned. A positive value is under budget, a negative value is over budget, and zero is on budget.
  - CV = $EV - AC$
- Cost Variance Percentage (CV%)
  - The percentage of labor hours the effort is under or over the value earned. A positive percentage is under budget, a negative percentage is over budget, and zero percent is on budget.
  - CV% = $\frac{CV}{EV} \times 100$
- Cost Performance Index (CPI)
  - An efficiency factor comparing costs relative to the value earned. A value greater than one is under budget, a value less than one is over budget, and a value of one is on budget.
  - CPI = $\frac{EV}{AC}$

So with respect to costs, Deliverable 1's status is as follows:

- Current month: CV = -100 hours, CV% = -12.5%, CPI = .89. The deliverable is 100 hours (i.e. 12.5%) over budget for the current month. The team accomplished .89 units of actual work for every 1 unit of spending for the current month.
- Cumulative: CV = -100 hours, CV% = -6.25%, CPI = .94. The deliverable is 100 hours (i.e. 6.25%) over budget for the project to date. The team accomplished .94 units of actual work for every 1 unit of spending on average.

2.2.4 Forecasting and Course-Correcting the Project

The project monitoring formulas in the previous section are only useful for conveying current state and cumulative progress. Fortunately, EVM offers project managers a predictive element as well. A forecast for the final project cost can be reached using three related formulas: *Estimate to Complete, Estimate at Completion,* and *Variance at Completion.* These formulas work for the entire project or for individual tasks.

The Estimate to Complete (ETC) is a measure of the remaining hours necessary to finish work on an effort. It can be reached in two ways. Informally, a project manager can examine a task and estimate how many hours are needed to complete the remaining EV. Formally, a project manager can estimate the remaining cost based on the project continuing at the observed cost performance level (CPI), for the remainder of the work ($BAC - EV_{CUM}$):

- $\text{ETC} = \frac{BAC - EV_{CUM}}{CPI}$

Neither the informal nor formal method is inherently correct. The informal method can be more accurate if the project manager has good information about the remaining work, but can also be less accurate if the project manager is not objective or does not predict obstacles. The formal method can be more accurate if the selected CPI truly reflects the efficiency going forward. CPI in this formula can be based on most recent month or the average over the project to date.

Once the task's ETC is calculated, the Estimate at Completion (EAC) is easily calculated as $ETC + AC_{CUM}$. The EAC is useful when shown in comparison to the BAC, which was determined at the start of the project. The Variance at Completion (VAC), $BAC - EAC$, is the project manager's best estimate of whether or not the effort will finish under (positive VAC), over (negative VAC) or on (zero VAC) budget.

The final EVM formula worth noting in this overview is known as the *To Complete Performance Index* (TCPI). TCPI is the cost efficiency factor necessary to achieve either the original planned budget (i.e. BAC) or the forecasted budget at completion (i.e. EAC). The formulas are respectively:

- $TCPI_{Planned} = \frac{BAC - EV_{CUM}}{BAC - AC_{CUM}}$
- $TCPI_{Estimated} = \frac{BAC - EV_{CUM}}{EAC - AC_{CUM}}$

TCPI in either of its forms can be compared to the effort's CPI. If the TCPI is greater than the CPI, the project manager must improve cost efficiency on the project if he or she wants to achieve the target budget. If the TCPI is less than or equal to the CPI, then the project manager stands a good chance of achieving the target budget.

This section has covered the fundamentals of defining and managing projects. The key takeaways from this section should be that projects can be organized into discrete quantifiable pieces, that progress and spending on each of these discrete pieces can be quantified and tracked, and that analytical tools are available to both predict and correct the path of a project. These fundamentals of project management will play an important role in developing the model proposed later in this thesis.

3 Organizations

A single individual or a small team of experts familiar with the project scope can perform most of the project definition described in Section 2. However, managing and executing all but the smallest of projects requires the support of one or more teams. These teams must collaborate with each other and the customer to ensure that the project will be a success. The combination of all teams working together to complete a specific project scope is referred to as the *organization*. This section is an overview of *organization structures* and their relationship to Earned Value Management.

3.1 Structuring Organizations

Forming and managing the right organization to achieve the project scope, within the allotted schedule and budget, is a project in its own right. The most obvious consideration for a project manager is to hire staff with the appropriate balance of skills for the project scope. But without the right framework in place, the project manager opens himself or herself up to gaps and overlaps in effort, degraded quality of product, and in the worst cases project failure or termination. The project's framework, or organization structure, is a command and communications network that links together a set of clearly defined teams.

Let us consider for a moment that every team member is a single node in a network. For N nodes, there are $N(N-1)$ possible uni-directional links between those nodes. Since each of these links has the potential to be either a command channel or a communication channel, there could be up to $2N(N-1)$ possible permutations of links between the nodes. As an example, a team of size N = 10 could be organized into any of 180 possible structures! Building and analyzing networks is a part of the field of Graph Theory, and more specifically Network Topology.

For the scope of this thesis, it is only important to note that teams and individuals can be considered as nodes in a network, and that the relationships between nodes can be defined as a combination of commanding and communicating. In reality, organizations are almost always broken up into teams and sub-teams to reduce the complexity of the command and communications network. So it is worth spending a moment to dive a little deeper into the most common team breakdowns used on large projects: function-oriented, product-oriented, and matrix.

3.1.1 Function-Oriented

A function-oriented organization has a single manager overseeing a set of teams, sometimes referred to as departments, each with expertise in one specific area (e.g. signal processing, software development, supply chain, etc). These teams are not tied to any one product or product line, but instead support all products that require their skill set. Teams can be decomposed further into niche areas of expertise within the larger skill domain. When working on only one product, a function-oriented organization resembles a standalone business in that it will have its own customers, staff, and financial reporting in addition to the teams that are focused on product development. When the organization

is handling multiple distinct products, the same management team and staff will be the point of contact for all customers, and will delegate work to the appropriate functional teams. It is up to the individual department leads to manage resources between the various projects they support.

The primary advantages of function-oriented organizations are as follows:

- Increases inter-product line coordination
- Decreases overhead in the interest of eliminating staff redundancy
- Avoids duplication of specialists and functional expertise
- Increases expertise in multiple product lines
- Facilitates multi-product integration and standardization
- Enables in depth knowledge and skill development

The primary disadvantages of function-oriented organizations are as follows:

- Decreases inter-functional coordination
- Obscures dependencies and handoffs between teams
- Obscures accountability from both the company and customer perspectives
- Worsens decision-making and ability to rapidly react to change
- Creates tunnel-vision with respect to larger corporate goals

3.1.2 Product-Oriented

A product-oriented organization has a single manager overseeing all teams involved in the lifecycle of one product or product line. Each team is focused on one stage of the product's lifecycle (e.g. design, development, integration, etc). These teams are further decomposed based on the constituent elements of the product as they apply to each team's charter. A product-oriented organization resembles a standalone business in that it will have its own customers, staff, and financial reporting in addition to the teams that are focused on the product development. This structure can be scaled in large organizations such that multiple product managers will report to a division manager who arbitrates resource issues and participates in strategic decision-making.

The primary advantages of product-oriented organizations are as follows:

- Increases inter-functional group coordination
- Clarifies dependencies and handoffs between teams
- Clarifies accountability from both the company and customer perspectives
- Simplifies decision-making and ability to rapidly react to change

The primary disadvantages of product-oriented organizations are as follows:

- Decreases inter-product line coordination
- Increases overhead in the interest of making each product team self-sufficient
- Duplicates specialists and functional expertise
- Increases specialization in only one product line
- Complicates multi-product integration and standardization

3.1.3 Matrix

A matrix organization is a hybrid of the function and product-oriented organization structures. In fact, a matrix organization actually is both a function and product-oriented organization existing simultaneously. The matrix has a single manager who oversees both a functional and a product manager. This matrix manager serves as the corporate point of contact, sets strategic goals for the organization, and arbitrates disputes between the functional and product managers. The functional and product managers have dedicated staff members, but the majority of matrix employees are members of both the functional and product organizations. This may require a little further explanation.

To understand the matrix organization, it is easiest to begin by picturing the leadership hierarchy beneath the product manager that would exist in a purely product-oriented organization (i.e. leads grouped by lifecycle phases). While these leads are accountable to the product manager, they are selected from and remain accountable to the functional manager as well. These matrix leads have complete control over forming their teams and sub-teams by drawing from other members of the functional organization. This product team, composed almost entirely of functional team members will remain in place until each phase of the project is completed, at which point the functional teams are dissolved and receive new product team assignments from their functional manager. The intent of this relationship is to maintain the function-oriented organization's strength of cross-product exchange of ideas, while maintaining the product-oriented organization's strength of clear handoffs and rapid decision-making.

The primary advantages of matrix organizations are as follows:

- Increases inter-product line coordination
- Increases inter-functional group coordination
- Avoids duplication of specialists and functional expertise
- Increases expertise in multiple product lines
- Facilitates multi-product integration and standardization
- Enables in depth knowledge and skill development
- Clarifies dependencies and handoffs between teams
- Clarifies accountability from both the company and customer perspectives
- Simplifies decision-making and ability to rapidly react to change

The primary disadvantages of matrix organizations are as follows:

- Increases overhead by maintaining both a functional and product organization simultaneously
- Creates competition between product teams for functional talent resources
- Creates conflicts of interest between employees' functional and product team loyalties

3.1.4 Discussion

It would be misleading to say that any one of the previously mentioned organization structures is outright superior to the others. In *The Organization and Architecture of Innovation*, Thomas J. Allen and Gunter Henn suggest that the appropriate organization structure for a project is tied to the nature of the innovation required by that project. "A simple depiction of innovation is one of a process that mediates between two streams of activity: the development of market needs and the development of technological capabilities or potential solutions to meet those market needs [1, pg 30]." So let us briefly reexamine each of these organization structures in the context of market needs and their associated technological requirements.

Function-oriented organizations are centered on domains of expertise and facilitate sharing of information between product lines. But these organizations are also slow to react to market pressures due to their lack of a strong, consolidated customer interface. This organization therefore is ideal for a project manager who wishes to promote intra-domain technological innovation, and is not at the mercy of near-term market needs.

Product-oriented organizations are centered on product lines and facilitate sharing of information between domains of expertise. But in these organizations, "specialists are separated from their knowledge base, and are less likely to stay informed. They focus almost exclusively on the peculiar aspects of their technology in the context of a particular project and soon lose sight of other applications and developments in that technology [1, pg 35]." This organization therefore is ideal for a project manager who wishes to promote inter-domain technological innovation and market responsiveness, and is willing to sacrifice long-term personnel and domain technological development.

Matrix organizations, like product-oriented organizations, are centered on product lines and facilitate sharing of information between domains of expertise. But since employees still maintain ties to their functional leads, and since employees rotate between product-oriented organizations, matrix organizations can also promote intra-domain technological innovation, and avoid sacrificing personnel domain knowledge. This type of structure would appear to have all the best, and none of the worst, characteristics of the function and product-oriented organizations. So regardless of market or technological pressures, would not a matrix style organization be ideal for every project manager?

There are two obvious disadvantages to a matrix organization. The first disadvantage is that the project manager will need to absorb the overhead associated with two distinct organizations instead of just one. Both the functional and product aspects of the matrix require their own management team as well as technical, financial, and new business staff. This personnel overhead is a significant financial burden to be taken on while arguably adding very little direct value to the customer deliverable. The second disadvantage is that competition is created over A-players and employees with unique functional skill sets. Pure product-oriented organizations suffer from this disadvantage as well. However, the situation is much worse in a matrix organization since they are designed to only retain staff for finite periods of time. This situation results in more

intense competition between individual matrix product managers who compete with each other to obtain the best staff on their next rotation.

There is one other reason for project managers to be wary of matrix organizations. It is both the strength and the weakness of this particular structure. A very old expression states, "No servant can serve two masters; for either he will hate the one and love the other, or else he will be devoted to one and despise the other." While biblical scholars estimate that this quotation was written around the first or second century AD, the author of this thesis is confident that the spirit of these words has applied since early man first organized to hunt game. And they apply just as much today when speaking of matrix organizations.

On paper, a matrix organization should create a positive tension between functional management and product management. "One force should be working to get the product out into the market; the other forces is holding back to guarantee product integrity [1, pg 41]." In practice however, and in the thesis author's personal experience, the matrix organization has a tendency to create conflicts of interest between employees' functional and product team loyalties. And from the management perspective, product managers tend to assume dominance over the matrix, as they are the organization's direct interface with the customer, while functional managers struggle to avoid the perception of merely serving a human resources role.

Hopefully, at this point it is clear that no one organization structure is inherently appropriate to align as a rule with a particular scope of work. Any organization structure can be made successful and this thesis has only covered three very common structures. The takeaway from this section should be that a project manager must choose a framework from which their staff can effectively command and communicate as they work to achieve the project scope. From this point, we can return once again to Earned Value Management.

3.2 Linking Structure to Scope and Funding

In Section 2, we introduced the concept of a Work Breakdown Structure (WBS), a hierarchical tree-structure that is organized around the primary deliverables of the project. The EVM methodology requires that project managers develop a WBS at the beginning of a project to ensure that all work within the project scope has been clearly defined and assigned unique ID numbers for monitoring purposes. Similarly, EVM requires project managers to capture their organization structure, regardless of the form it takes, to ensure that all personnel resources that will be assigned to the project scope have been clearly defined, and to ensure that reporting relationships are well understood. In EVM terms, this concept is referred to as an *organizational breakdown structure* (OBS).

3.2.1 Organizational Breakdown and Responsibility Assignment

An OBS is a hierarchical tree-structure that is organized around the reporting relationships of the organization. On most projects, the OBS is simply referred to as the "org chart" and for all intents and purposes of the two are the same. The primary uses of the OBS are to group personnel into teams, to communicate authority over those teams,

and to resolve disputes between teams. The OBS additionally serves as the basis for assigning organizational accountability to major elements of the project scope defined in the WBS. When the OBS is combined properly with the WBS, the project manager gains assurance that all elements of the project scope have been assigned to an accountable element of the organization. This combination of WBS and OBS is referred to as the *responsibility assignment matrix* (RAM).

The RAM is a tool that maps the project WBS to the project OBS. Both *responsibility* and *accountability* can be captured explicitly where elements of the WBS and OBS intersect, as portrayed in Figure 3.2-1 below.

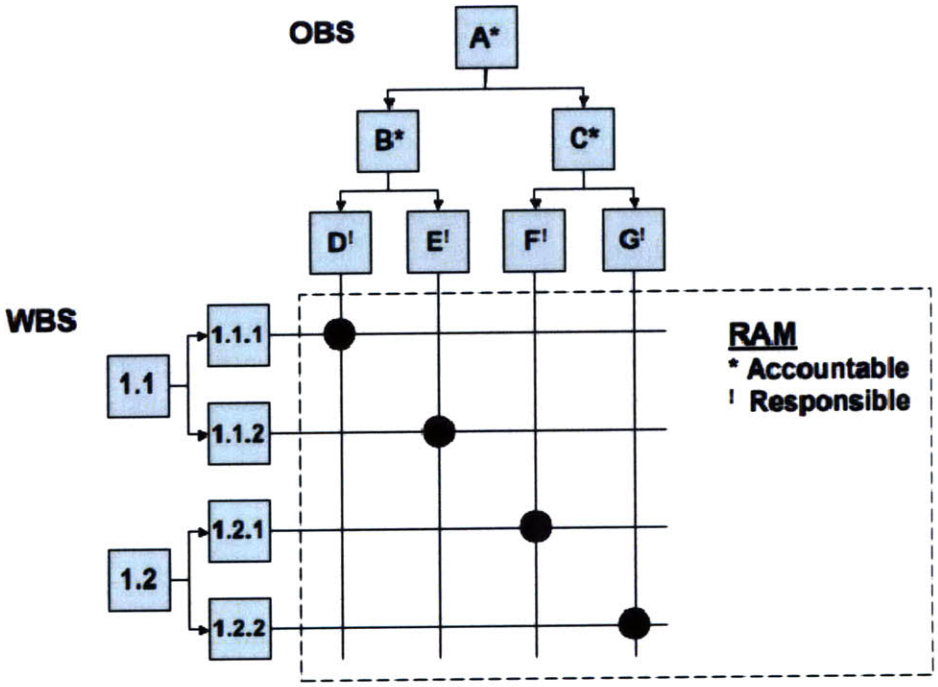

Figure 3.2-1 RAM

For the purposes of this discussion, a *responsible* individual is one who expends effort to complete a task and there can be multiple *responsible* individuals per element of project scope. In contrast, an *accountable* individual is one who the project manager has entrusted to monitor and deliver an element of project scope. *Accountable* individuals oversee the work of *responsible* individuals and there can be only one *accountable* individual for a given element of project scope. The RAM can be maintained in either tabular or graphical form, so long as the one-to-one mapping of accountability to scope is clear.

3.2.2 Breaking Down Contract Funds

Continuing with our example project, let us say that a customer has just awarded our project manager a contract. Assuming our manager is a part of a larger corporation, a portion of this money may be taken away immediately as profit. In EVM terms, the remainder is referred to as the *total allocated budget* (TAB). This is literally the total dollar amount available to the project manager to complete the scope. Next, the project manager sets aside some portion of the TAB in reserve for emergency situations such as unforeseen scope or research needs. In EVM terms, this remainder is referred to as the *performance measurement baseline* (PMB). It is the PMB that is the highest-level summary for all reporting on the project's progress.

The PMB can be thought of as consisting of two types of funds: planned and unplanned. The PMB's planned funds are known as *control accounts*; the primary components that will be monitored on the project. *Planned* in this context means that the funds have been dedicated to a portion of the project scope and have been assigned to an accountable individual...a *control account manager* (CAM). Control accounts are decomposed one step further into *work packages* and *planning packages*. Work packages are scheduled, funded elements of scope, which are either actively being worked or will be imminently. Planning packages are also scheduled and funded elements of scope, but they haven not yet been assigned milestones and will not be executed until some point in the future. Together, work packages and planning packages are the lowest level of monitoring on the project. The last remaining aspect of the PMB is its unplanned funds, or *undistributed budget*. These funds are any remaining money that the project manager has yet to dedicate to elements of scope. Figure 3.2-2 below depicts the contract funds breakdown.

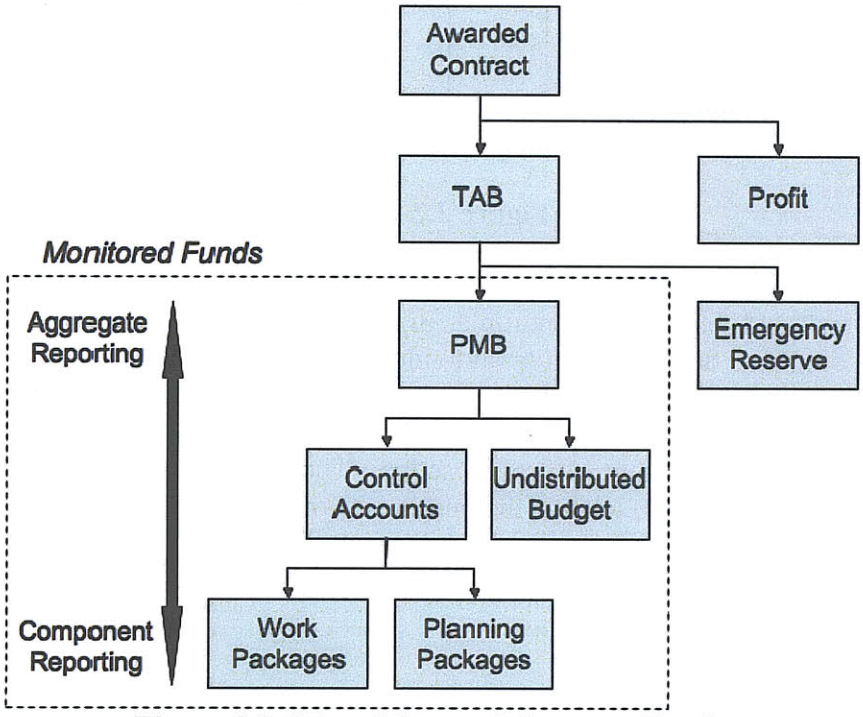

Figure 3.2-2 Breakdown of Contract Funds

This section has explained the concept of organization structure and its relationships with the elements of project scope and contract funding. It should be clear at this point that a project manager must be equally concerned with properly aligning personnel, scope, and funding as he or she is with planning and executing the project itself. From the perspective of this thesis, one of the key ideas introduced in this section is that of control accounts. In Section 2.1.4, project management was illustrated as an exercise in balancing the interaction of the three pillars of project management: scope, schedule, and budget. Control accounts are the intersection of these three pillars, and the success or failure of a project depends on a network of control account managers and their ability to command and monitor their assigned work. The nature of control and control systems will be explored in the next section.

4 Control Systems

To control something is to command or exert influence over its behaviors. When processes are left uncontrolled, disturbances may cause an otherwise stable system to behave undesirably. In the interest of preventing or mitigating negative system impact, governing devices or *control systems* are put in place to monitor and affect the behaviors of another system's processes. The engineering discipline that studies the design and analysis of methods of controlling systems is known as Control Theory.

The classic introductory example of a control system is a thermostat. In this situation, the thermostat monitors the temperature of a room and activates heating or cooling elements whenever the room's temperature becomes higher or lower than a set temperature. The *controlled variable* in this example is the temperature of the room. Disturbances such as the opening and closing of windows can cause a room's temperature to fluctuate. The thermostat control system is put in place to control a single variable within the room system. One can imagine another control system being put in place to control the lighting level of the room. Here, the intensity of light bulbs in the room could be raised or lowered as disturbances such as changing amounts of sunlight cause the room's lighting level to vary.

Control systems are all around us and in some cases humans actually are a part of the control system. Consider the level of fuel in the gas tank of a car. As the car's engine consumes fuel, the car's gas gauge indicates declining fuel levels. Once the warning light is illuminated (i.e. when the fuel level crosses below a minimum threshold), it is up to the operator to refuel the vehicle if they wish to continue using the car. In this situation, the human is the control system that monitors the fuel level and acts upon a visual signal from the gauge to affect the state of the fuel tank.

With these examples in mind, let us now walk through the fundamental components in a control system.

4.1 Control System Components

There are three principle component types within any control system: a controller, a sensor, and an actuator. In order to successfully control a system, there must be at least one component of each type, but there can be many more of each if necessary. As a note, control systems come in a myriad of combinations of these components. This brief introduction to the component types explains the general form of each component, as well as the components' most common inputs and outputs. The section closes with the three most common integrated forms of control systems.

4.1.1 Controller

The controller component type is the decision-maker for the control system. Its primary function is to issue commands, based on input data, which will affect the controlled process. Figure 4.1-1 below depicts the general form of a controller component. At the center of the controller is the controlled process model, one of the most important aspects of the entire control system. This model is the control system's internal representation of a real-world process and we will examine it further in a moment.

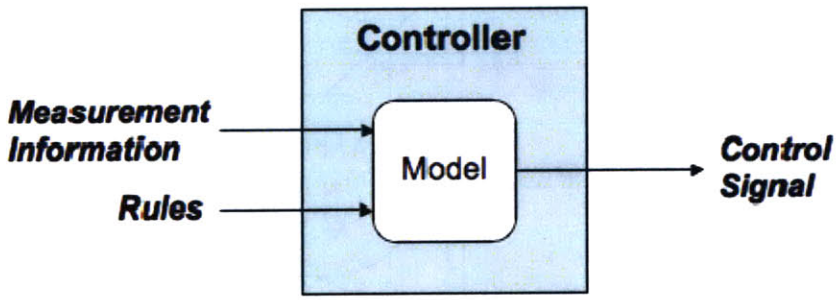

Figure 4.1-1 Controller Component

There are two kinds of controller input data portrayed in Figure 4.1-1: *measurement information* and *rules*. There is also one kind of output data from the controller known as the *control signal*. Let us put these terms in the context of a digital thermostat's controller, a very basic computer. As a single-purpose controller, the only characteristic of the room that is of interest to the thermostat computer is the air temperature (i.e. the measurement information). The computer's charter is to determine when to send activation or deactivation commands (i.e. control signals) to the room's heating and cooling system, based on a user-selected desired temperature (i.e. rules). In this simple example, the only input rule is a temperature set point. The computer is designed to understand that this input means "maintain this temperature at all times." In more complex examples, such as a programmable thermostat, the input rules can be conditional statements that pair desired temperatures with the day of the week and time ranges. It should also be noted that rules could be built into the controller itself, so that the measurement information is the only necessary controller input.

An important point to consider is that the control system's ability to successfully govern is largely dependent on having an accurate internal model of the controlled process. This model can be represented as a finite state machine (FSM) such as Figure 4.1-2 below. In this model, the room is always in exactly one of three possible states: "Correct Value," "Above Value" or "Below Value." A complete FSM will define all possible states of a system and account for all transition events that force the system to go from one state to another. In FSM notation, system states are represented as circles and transitions are represented as directed arrows. Each transition is labeled with an event that will cause a state transition and one or more actions that the controller must take to complete the transition. This is presented in shorthand as "[Event Description] / [Controller Action]."

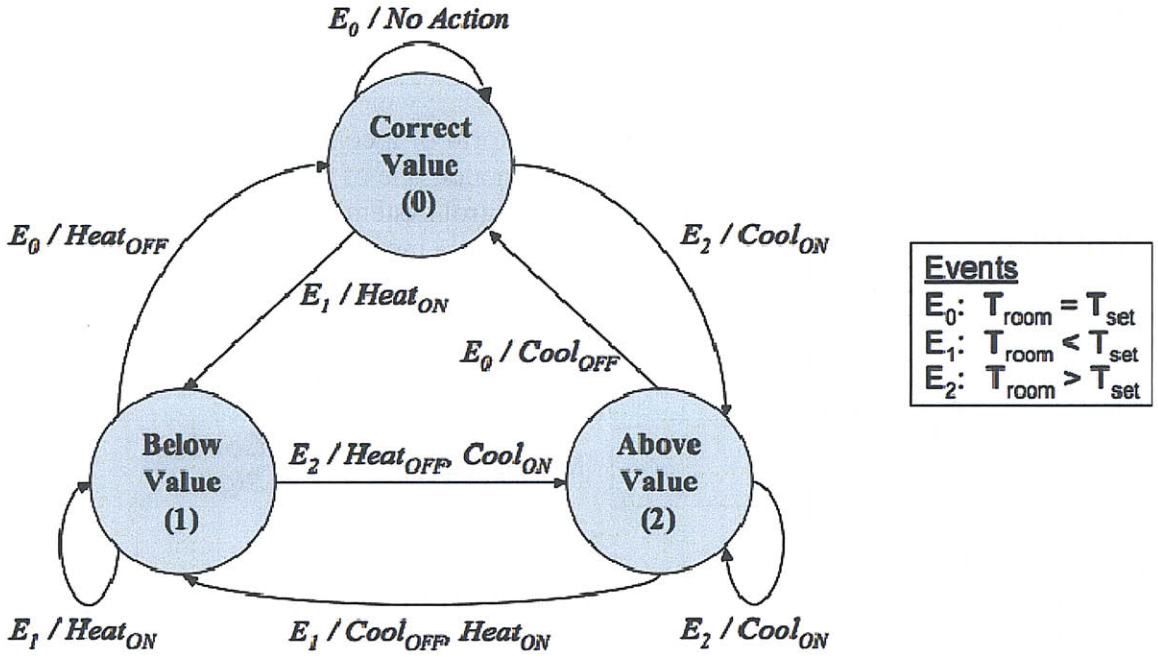

Figure 4.1-2 Controller Finite State Machine

Next let us explore the source of the controller's measurement information, the sensor component type.

4.1.2 Sensor

The sensor component type is literally the sensory organ of the control system. Its primary function is to act as the controller's source for information on the current state of the controlled process. Figure 4.1-3 below depicts the general form of a sensor component. At the center of the sensor is a converter, a device that takes the observed system data and alters it into useful information that the controller can act upon.

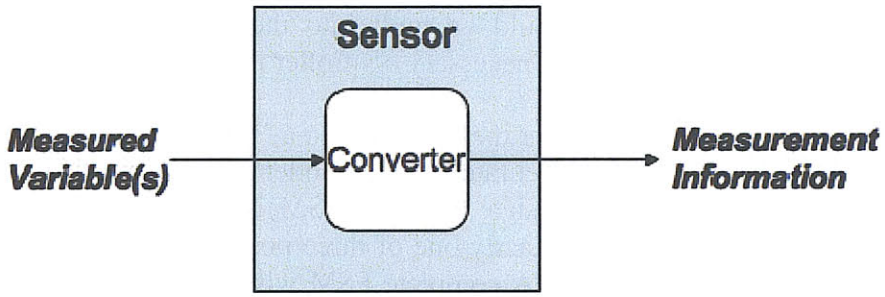

Figure 4.1-3 Sensor Component

The distinction between measured variables and measurement information may be unclear. To understand these terms better, let us return to the digital thermostat example. Digital thermostats contain a single thermometer sensor, which in turn contains two converters. The first converter is called a thermistor, a device that varies its resistance

with changes in temperature. In this case, the measurement information is the voltage across the thermistor that can be directly interpreted by an electronic controller. The measured variable, as you would expect, is the heat from the room air as it transfers in and out of the thermistor. Thus the measured variable, heat, is converted by the thermistor's temperature-based changes in resistance into measurement information, voltage.

As previously mentioned, there are two converters in a digital thermostat. One feeds measurement information (i.e. voltages) to the electronic controller that will issue command signals to the room's heating and cooling system. The other converter, a display, takes the same measurement information and changes voltages into text for the human that can read the current temperature and set the desired temperature for the thermostat computer.

Now, let us explore the recipient of the controller's signals, the actuator component type.

4.1.3 Actuator

The actuator component type is the manipulator of the control system. Its primary function is to act upon the control signals it receives from the controller, in the interest of affecting changes in the behavior or state of the controlled process. Figure 4.1-4 below depicts the general form of an actuator component. Similar to the sensor component, at the center of the actuator is a converter that takes desired control actions and alters them into a form that can impact the controlled process.

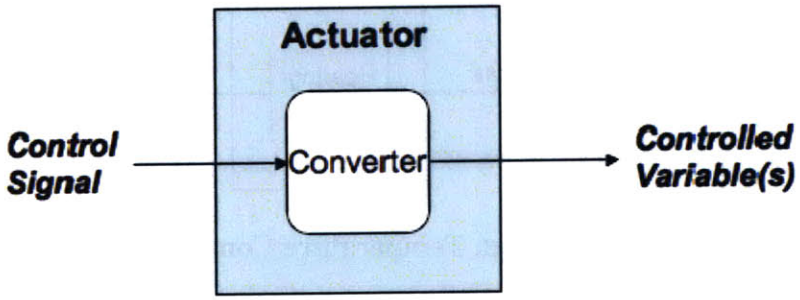

Figure 4.1-4 Actuator Component

Some control systems are so basic that they require only one actuator. The lighting system found in any room is a good example of this design. A person may determine that the room they are in has become too dark and may decide to flip a switch to turn on the room's light. In this example, the human is the controller and flipping the switch to the "on" position is the control signal for the light bulb actuator to emit light. The control signal is an electrical current, which a filament in the light bulb converts to heat energy thus releasing visible light, the controlled variable.

The thermostat control system example is the same in concept as the room lighting example, although several actuators are required to perform the conversion from control signal to controlled variable. In a house with an oil-burning furnace, the thermostat

controller sends a control signal to a fuel pump to draw oil from a tank. This oil is sprayed into a burner, which mixes with air and ignites in a chamber. The chamber increases in temperature and heats either ventilated air or circulated water. The air or water is pumped throughout the house, thus ultimately raising the room temperature. In this example, both the fuel pump and burner are actuators. The fuel pump converts electrical signals into oil flow and the burner converts oil flow into heat energy.

The actuator is the last major component type in a control system. The next section explores the integration of all three of the component types.

4.1.4 Integrating Control System Components

In the previous sections, the individual components of a room temperature control system were presented. Figure 4.1-5 below shows the fully integrated form of this control system in the context of its operating environment.

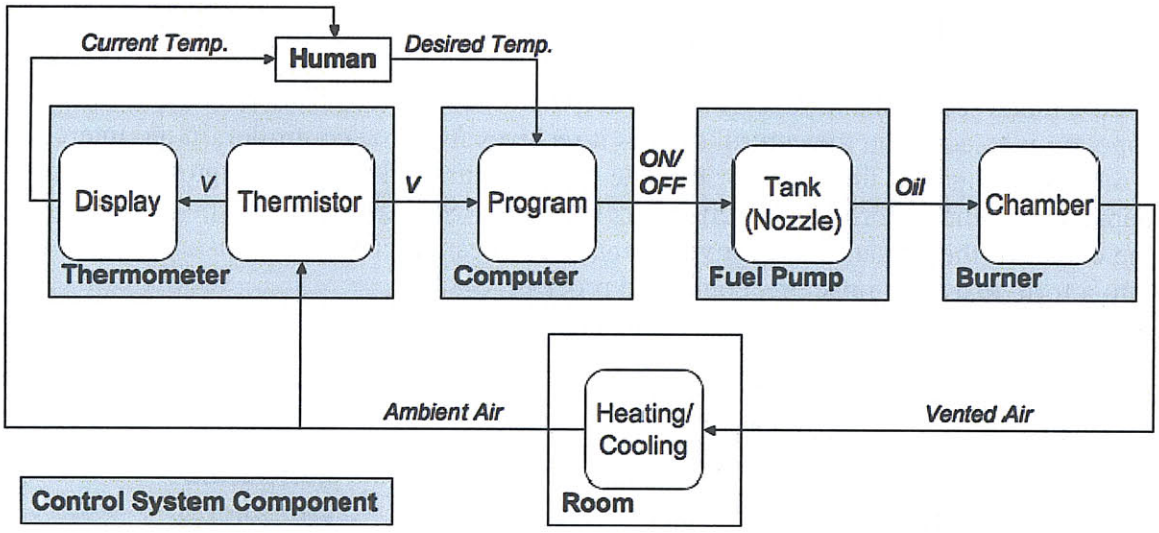

Figure 4.1-5 Room Temperature Control System

In examining the above diagram, the reader may have noticed that the human operator is shown as an entity external to the room temperature control system (i.e. not a control system component). The diagram was drawn this way for the purpose of clearly illustrating how component types can be integrated, but in fact this depiction is not quite correct. The human is as much a part of the control system as the thermostat itself. From the perspective of the computer, the human is only a source for the input rules that govern the program model. But in the larger operating context, the human is actually an entirely separate, self-contained control system. The human is able to sense the ambient air temperature, to associate the sensed temperature with desirable and undesirable states in a mental model, and to issue commands that will affect a controlled process. If drawn accurately, the human would be depicted as a control system for the control system. Together, these nested control systems affect change in the room's heating and cooling process. The concept of hierarchical control systems is central to this thesis and will be explored further in a later section.

One additional point to note about Figure 4.1-5 is that the room itself is not being controlled by the control system. This control system was put in place solely to control one process occurring in the room: the heating and cooling of the room's air. This may seem like an obvious point, but to design a successful control system one must properly align the process to be controlled, the measured and controlled variables, and the controller model. Correct alignment within the control system is one of the key criteria for a successful design. This criterion and several others will be explored in a later section, but let us spend a moment discussing the alignment criteria in the context of the room temperature control system.

In Figure 4.1-5, the controlled process is correctly defined as "Heating/Cooling." The controlled variable for this process is "Vented Air," yet the measured variable for this process is shown to be "Ambient Air." This may appear to be a discrepancy at first, but it is actually a critical part of the control system design. If the measured variable was "Vented Air" like the controlled variable, then the controller would essentially be comparing the temperature of the air as it leaves the oil burner to the "Desired Temperature," when the logical intention of the human operator was to have the air in the room maintained at a desired temperature. Thermostats are physically separated from air vents and radiators for this exact reason. If the thermostat were incorrectly placed, the controller would erroneously receive sensor data on the vented air. This failure would be unrecoverable, since the controller model has no concept of rooms, air or even humans for that matter. As was shown previously in Figure 4.1-2, the states of the controller model are merely designed to know if the sensor data is at, above, or below a specified setting. This in brief, is why proper alignment is so critical to the success of a control system.

The room temperature control system example we have been discussing in the past few sections is commonly referred to as an "On-Off Feedback" control system design. The next section explains these terms and discusses other common designs of control systems.

4.2 Control Systems Designs

Control system designs can generally be classified into two categories: non-feedback and feedback. Feedback is the controller's means of updating its state model of the controlled process. In the room temperature control system example, feedback is present in the form of the thermometer sensor taking continuous measurements of the ambient air and relaying the information back to the controller to make decisions. Without this information, the controller would have no confirmation of the impact its command signals were having on the controlled process. The presence or absence of feedback is fundamental to the design of a control system, thus it is the primary means by which control systems are categorized. This section describes the two basic categories of control systems, their uses, and their advantages and disadvantages.

4.2.1 Non-Feedback Control Systems

Non-feedback control systems, also sometimes referred to as "open-loop" control systems, have no sensor component and thus no means of flowing information on the controlled process back to the controller. In order to be a successful design, a non-feedback control system must contain an extremely accurate model of the controlled process and all possible disturbances. Non-feedback control systems could also be successful without an accurate internal model, if the state of the controlled process is unimportant to the controller's actions.

It may not be obvious why one would intentionally design a control system to be ignorant of its impact on the controlled process. But when designing any system, there are always tradeoffs between functionality and cost that must be considered. Adding sensors and the necessary controller functionality to interpret their output can be expensive. This financial cost must be balanced against the level of control needed for the controlled process as well as the impact to the system and its environment if the process is not precisely controlled. Some common examples of non-feedback control systems include: kitchen microwaves, lawn irrigation systems and clothes dryers.

The advantages of non-feedback control system are that they are inexpensive and are sufficient for overseeing processes that have no disturbances or have very well defined disturbances. The disadvantages of a non-feedback control system are that they cannot react to any changes in the controlled process or its environment and that they have no ability to recover from failures.

4.2.2 Feedback Control Systems

Feedback control systems, also sometimes referred to as "closed-loop" control systems, have one or more sensor components flowing information on the controlled process back to the controller. In order to be a successful design, a feedback control system must contain an accurate model of the controlled process and rules for dealing with disturbances should they occur. Feedback control systems offer precise control over the state of the controlled process and can recover from failure states if designed properly.

The room temperature control example used throughout Section 4.1 is just one common example of a feedback control system. Another common example is the flush-refill control system in toilets. Here, a valve actuator is opened and closed based on information provided by a floating ball "sensor/controller" in the tank. The floating ball is a feedback mechanism that measures water level and physically signals the valve to close when a pre-determined level is reached. A more complex example of a feedback control system would be the autopilot found in most commercial aircraft. Here, hundreds of sensors and advanced controllers are used to command hundreds of actuators in the interest of keeping the aircraft at a set speed, altitude, and course.

The advantages of a feedback control system are that they offer precise control, can react to changes in the controlled process and its environment, and have the ability to recover from failures. The disadvantage of feedback control systems is that they can become expensive as sensors are integrated with the controller. Also, the complexity of a

feedback-based controller is likely to increase the overall system cost. Controller complexity is directly dependent on the level of control the designer wishes to have over the system's reaction to disturbances. There are many different types of feedback-based controllers. Let us look at the four most common of these feedback sub-categories briefly.

4.2.2.1 On-Off Control

An "On-Off" controller is the simplest means of controlling an actuator. Its control signal is always a step function, meaning a binary command to the actuator to either turn on or turn off. If the signal is "On," the actuator is being commanded to perform its function at a fixed value. So in the case of the thermostat, an "On" command to the heating system is asking the heater to turn on and produce its maximum possible output immediately. When the thermostat eventually issues the "Off" signal, the heater is being commanded to cease functioning entirely. On-Off controllers issue no command signals between "Off" and maximum. The mathematical expression of an On-Off thermostat controller is:

- IF $T_{Room} > T_{Set}$, THEN $Signal = 0$; ELSE $Signal = 1$
  - Where T_{Room} is the most recent ambient air temperature,
  - T_{Set} is the desired ambient room temperature, and
  - $Signal = 1$ is understood by the heater to mean maximum output

On-Off feedback control systems are easy to design and analyze. And they are cheaper to build than other feedback-based systems because the sensors and actuators involved require less accuracy. However, On-Off controllers are inefficient, because they force the actuator to always operate at maximum or not at all. This controller type does not converge on the set point; it continuously oscillates within a band centered on the set point. Also, the fluctuations around the set point that inherently result from using an On-Off controller can result in accidents if the positive or negative overshoots are too drastic.

On-Off feedback control systems do offer more precise control than non-feedback systems, but feedback controllers that incorporate gain into their command signals are even more precise. The three basic gain controllers are: *Proportional, Integral, and Derivative.*

4.2.2.2 Proportional Control

A proportional or "P" controller is a basic method of issuing continuous (as opposed to binary) commands to an actuator. These commands allow the actuator to output values other than just zero or maximum. The goal of a P-controller is to issue higher signal values when major differences between current and desired measurement variables occur, and to issue smaller signal values when minor differences between current and desired measurement variables occur. This is literally a proportional signal response. There are two calibration constants associated with P-controllers. The first is the *proportional gain*, a constant used to adjust the time necessary for the controlled variable to reach steady state. The second is the *calibration offset*, a constant used to tune the output signal when

the difference between current and desired measurement variables is zero. The mathematical expression of a P-controller thermostat is:

- $Signal = K_C(T_{Set} - T_{Room}) + C$
  - Where T_{Room} is the most recent ambient air temperature,
  - T_{Set} is the desired ambient room temperature,
  - K_C is the proportional gain constant, and
  - C is the calibration offset from the set point

P-controller systems are simple to design and analyze, easy to calibrate, provide good stability and demonstrate fast conversion times. Unfortunately, P-controllers also exhibit two negative qualities. First, the controller must remain constantly active to maintain equilibrium, which cannot occur if the measured variable is at the set point. Thus the calibration offset is necessary. Unfortunately, the introduction of the offset means that at equilibrium, the P-controller signal will converge on the offset value, not the set point. The second shortcoming of P-controllers is in the gain constant. Larger values of the gain constant will bring the controller to steady state more rapidly, but will also cause system instability if the value is too high. The amount of oscillation resulting from higher gain constants may be unsuitable for some control applications.

4.2.2.3 Integral Control

An integral or "I" controller is another method of issuing continuous commands to an actuator. The signals of I-controllers, like those of the P-controller, are proportional to the measured error in the controlled process. However, I-controllers differ from P-controllers in that they consider the history of past measurement errors, as opposed to looking solely at the current measurement error. There are two tunable parameters in an I-controller. The first parameter is the *integral gain*, a calibration constant used to adjust the time necessary for the controlled variable to reach steady state. The other tunable aspect of the I-controller is the time window (i.e. integral interval) over which the controller will integrate. The mathematical expression of an I-controller thermostat is:

- $Signal = \tau_I \int_0^{\tau} (T_{Set} - T_{Room}) dt$
  - Where T_{Room} is the most recent ambient air temperature,
  - T_{Set} is the desired ambient room temperature, and
  - τ_I is the integral gain constant

I-controllers are easy to calibrate, provide good stability, resist noisy input signals and unlike P-controllers they will eventually converge on the set point. Unfortunately, I-controllers also exhibit two negative qualities. First, lower values for the integral gain constant will result in faster set point conversion, but will also increase oscillation and possibly lead to system instability. Higher values for the integral gain constant will result in longer conversion times, but will resist oscillation. The second drawback of an I-controller is in determining the size of the integration window. A large integration window will result in faster conversion times, but may result in significant overshoots, as

the controller will be biased by persisted noisy inputs. A small integration window will minimize overshoots, but the smaller the window, the closer the I-controller is to becoming a P-controller.

P and I-controllers can be combined into a PI-controller in the interest of capitalizing on the strengths of both. The mathematical expression of a PI-controller thermostat is:

- $Signal = K_C(T_{Set} - T_{Room}) + \tau_I \int_0^{\tau} (T_{Set} - T_{Room})dt + C$

PI-controllers are successful at eliminating the calibration offset found in P-controllers, and have the set point convergence and noise-resistant properties of I-controllers. But PI-controllers also have worse convergence response times and oscillation periods than pure P-controllers.

4.2.2.4 Derivative Control

The last gain controller type is a derivative or "D" controller. Just like P and I-controllers, D-controllers issue continuous actuator commands proportional to the measured error in the controlled process. However, D-controllers differ from both P and I-controllers in that they consider the current rate of measurement error, as opposed to looking at the magnitude of the current or historical measurement error. There is only one tunable parameter in a D-controller. The *derivative gain* is a calibration constant used to adjust the time necessary for the controlled variable to reach steady state. The mathematical expression of a D-controller thermostat is:

- $Signal = \tau_D \frac{d(T_{Set} - T_{Room})}{dt}$
  - Where T_{Room} is the most recent ambient air temperature,
  - T_{Set} is the desired ambient room temperature, and
  - τ_D is the derivative gain constant

D-controllers are easy to calibrate and resistant to constant changes. Unfortunately, D-controllers are also very sensitive to input spikes, and they never converge on the set point. D-controllers are used when trying to keep the output signal steady at whatever point it is currently at, as opposed to trying to bring the signal back to a set point. When calibrating the derivative gain constant, larger values will decrease overshoot, but may lead to unrecoverable instability due to magnification of signal spikes.

Just like P and I-controllers, P and D-controllers can be combined into a PD-controller in the interest of capitalizing on the strengths of both. The mathematical expression of a PD-controller thermostat is:

- $Signal = K_C(T_{Set} - T_{Room}) + \tau_D \frac{d(T_{Set} - T_{Room})}{dt} + C$

PD-controllers have some advantages over a pure P-controller. Namely, they have less overshoot and oscillation due to the D-controller property of resisting change. A disadvantage of the PD-controller is that with change resistance comes a slower response time than a P-controller alone. Additionally, PD-controllers still have the P-controller property of the calibration offset, as well as the disadvantage of the D-controller settling on points other than the set point.

As one may expect, P, I, and D-controllers can also be integrated into a single PID-controller. This form of controller has the best properties of all three individual gain controllers and virtually eliminates their characteristic weaknesses. The PID-controller issues signals that consider the past, present, and predicted future errors between measured variable and desired set point. This controller type can also be tuned, through the gain constants, to balance or emphasize the individual properties of its P, I, and D components. The disadvantages of using a PID-controller are that it is typically more complex, expensive, and difficult to analyze than any of the other controller combinations. The mathematical expression of a PID-controller thermostat is:

- $$Signal = K_C(T_{Set} - T_{Room}) + \tau_I \int_0^{\tau} (T_{Set} - T_{Room})dt + \tau_D \frac{d(T_{Set} - T_{Room})}{dt} + C$$

Each control system concept introduced in this section has advantages and disadvantages. The purpose of this section was not to propose one control system design over another, but instead to illustrate cost and functionality tradeoffs to be considered.

4.3 Control Systems and Error

At the beginning of Section 4, it was explained that processes can exhibit undesirable behaviors when left uncontrolled. One cause of undesirable behaviors was stated to be *disturbances*, influences external to both the control system and the process in question that transition the process from a favorable state to an unfavorable one. But disturbances are not the only cause of systems behaving undesirably. Other possible causes could be bad inputs to the process or even inherent flaws in the process itself. Ironically, the very control system that is put in place to prevent or mitigate negative system behaviors can itself be the cause of controlled process issues. These control system-induced errors can be introduced into a process via the controlled variable. So having discussed the fundamentals of control system components and control system designs, this last section briefly discusses the causes of errors that occur within control systems and the controlled processes they oversee.

Generally speaking, there are three categories of error that can manifest within a control system: internal issues, interface issues, and performance issues. An internal issue is a structural or behavioral design property of the component in question that will lead that component to generate an undesired output. An interface issue is a failure or absence of a screening mechanism for invalid input data. A performance issue is a failure to produce expected results in time or at the rate they are needed. Let us examine specific sources of error in the context of the overall control system.

4.3.1 Controlled Process Errors

Section 4.1 presented the general form of each major component of a control system. For completeness, Figure 4.3-1 below depicts the general form of a controlled process.

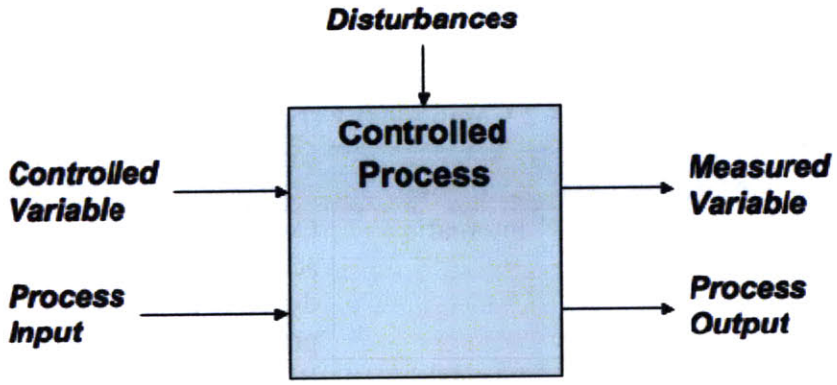

Figure 4.3-1 Controlled Process

The definitions of controlled variables, measured variables, and disturbances should be clear at this point. However, two new terms are introduced in Figure 4.3-1: *process input and process output*. Respectively, these two terms represent the inputs that feed the controlled process and the resulting process outputs. Note that disturbances, process inputs, and process outputs are a part of the controlled process regardless of the presence or absence of a control system. Controlled processes have four possible causes of error. These causes and their descriptions are captured in Table 4.3-1 below:

Table 4.3-1 Controlled Process Error Sources

Cause of Error	Issue Category	Description
Innate process flaws	Internal	Process does not account for all possible states or transitions; process transitions to undesirable state upon valid input data
Unscreened process inputs	Interface	Invalid process input data is accepted through a valid channel, due to lack of verification mechanism
Unscreened controlled variables	Interface	Invalid controlled variables are accepted through a valid channel, due to lack of verification mechanism
Unmitigated disturbances	Interface	External influences transition the process to a terminal failure state, due to lack of mechanism for self-correcting

4.3.2 Component Errors

A perfectly designed control system would not introduce any error into the controlled process. Unfortunately, it is very difficult to design a perfect control system. Controllers, sensors, and actuators all have unique issues that can cause undesirable behaviors in the controlled process if not designed properly. These causes and their descriptions are captured in Table 4.3-2 below:

Table 4.3-2 Component Error Sources

Component Type	Cause of Error	Issue Category	Description
Controller	State model flaws	Internal	Controller's state model does not accurately represent static or dynamic aspects of the controlled process
Controller	Poorly calibrated tuning parameters	Internal	Bad values for adjustable constants found in proportional gain controllers lead to control signal overshoot or oscillation
Controller	Unscreened measurement information	Interface	Invalid measurement information is accepted through a valid channel, due to lack of verification mechanism
Controller	Unscreened settings or rules	Interface	Invalid settings or rules are accepted through a valid channel, due to lack of verification mechanism
Controller	Response time	Performance	Signals not issued to actuators in time to affect controlled process
Sensor	Converter flaws	Internal	Faulty translation of measured variable to measurement information
Sensor	Unscreened measured variables	Interface	Invalid measured variables are accepted through a valid channel, due to lack of verification mechanism
Sensor	Response time	Performance	Measurement information not provided to controller in time, or measured variable sampling rate too low
Actuator	Converter flaws	Internal	Faulty translation of measured variable to measurement information
Actuator	Unscreened control signals	Interface	Invalid control signals are accepted through a valid channel, due to lack of verification mechanism
Actuator	Response time	Performance	Controlled variable not introduced in time to affect controlled process

4.3.3 Component Interface Errors

Any of the component types within a control system can interface with one another. At a minimum, a successful control system will have at least one instance of each of the following interface types: controller-actuator, actuator-process, process-sensor, and sensor controller. It is common in a complex control system for a component of one type to interface with many components of a different type (e.g. one controller receiving measurement information from multiple sensors) and for two or more components of the same time to interface with each other (e.g. two controllers exchanging measurement information with each other before issuing control signals). Component interfaces have three possible causes of error. These causes and their descriptions are captured in Table 4.3-3 below:

Table 4.3-3 Component Interface Error Sources

Cause of Error	Issue Category	Description
Channel corruption	Internal	Interface introduces signal noise or loss that corrupts otherwise valid information exchange
Incorrect connectivity	Interface	Interface incorrectly links or fails to link two or more components together, resulting in gaps or overlaps in information exchange
Response time	Interface	Interface introduces delays in information exchange between two or more components.

4.3.4 Closing Thoughts

This section has introduced the reader to control system components, their common integrated forms, and their causes of failure. These descriptions have intentionally been kept very generic. Every control system is unique and every component type has hundreds if not thousands of specialized forms. It is critical for the designer of a control system to study the process to be controlled to determine the right control system for the job. It is also critical for the designer to consider cost and functionality tradeoffs, as not every controlled process requires the same level of error control and system stability.

There are two final items to note before closing out the topic of control systems. First, not every process can be controlled. Table 4.3-1 listed the general forms that controlled process error can take. But even if a process is internally stable, screens for bad inputs, and is resistant to external disturbances, a control system may not be able to be put in place. From the perspective of Control Theory, a process can only be controlled if its states can be affected by a controlled variable and if the impact of changes to a controlled variable can be externally observed. The second item to note is that even if every component in a control system is behaving properly, the effects of improper interaction between the components may still lead to an overall system failure [4].

Having introduced the fundamentals of project definition and management, organizations, and control systems, it is now time to integrate these concepts into a cohesive framework to reduce programmatic risk on complex systems projects.

5 Program Framework

The term *programmatic risk* has been used several times in this thesis, but what exactly does it mean? Recall from Section 2.1 that a project was defined to be a series of interrelated tasks performed to meet one or more explicit goals (i.e. scope). Also recall that all projects have constraints (e.g. schedule and budget) associated with achieving the explicit goal. These constraints become the implicit goals of the project. A potential risk is present if any of the project goals, explicit or implicit, are in danger of not being accomplished. Programmatic risk expands this definition to also include characteristics of the organization that is executing the project, which may inhibit the successful completion of project goals. When risks are realized, a programmatic loss occurs. That is to say the program as a system has failed if the project scope is not achieved, or if the project scope is achieved, but the project's budget and schedule constraints were violated in the process. The prevention and mitigation of programmatic risks, in the interest of avoiding programmatic losses, is the purpose of this thesis.

Sections 2, 3, and 4 of this thesis have introduced topics that are central to understanding the nature of programmatic risk. In Section 2, "Projects and Project Management," the topics of project definition and Earned Value Management were presented. In Section 3, "Organizations," the topic of organization structure and its relationship to project scope and funding sources were presented. In Section 4, "Control Systems," the topics of control systems components, designs, and sources of error, were presented. This section of the thesis opens by explaining the relationships between the topics presented in sections 2, 3, and 4. It goes on to develop a specification for a structural framework to control programmatic risk and losses. The section closes with a proposed design for a control system that meets the developed specification.

5.1 Framework Relationships

Project management, *organization structures,* and *control systems* are three subjects that are deeply intertwined. Project management is the discipline of planning and managing resources to bring about the successful completion of a project's goals. An organization is a command and communication structure where teams work together to complete a specific project's scope. Control systems are governing devices that are put in place to monitor and affect the behaviors of another system's processes. A common pattern can be found in all three of these subjects. In each case, high-level objects are defined, interfaces are established between them, and they are then decomposed into a hierarchy of lower-level objects. From the project management perspective, this pattern is reflected in the development of a project plan and the WBS. From the organization perspective, this concept is reflected in the development of the OBS and the breakdown of contract funding. And from the control systems perspective, this concept is reflected in the design of individual and hierarchical control systems.

The existence of a relationship between these subjects is fairly obvious, but let us examine specific instances of where these subjects intersect in the interest of forming a solid foundation for the proposed thesis framework.

5.1.1 Command Relationships

In this thesis framework, controllers are equivalent to managers and actuators are equivalent to that manager's team members. In Section 4.1.1, controllers were defined to be decision-makers that issue control signals based on input data, in the interest of controlling a process. In Section 4.1.3, actuators were defined to be manipulators that affect changes in the behavior or state of a process, in response to control signals received from a controller. This relationship between a controller and an actuator is identical to the relationship between a manager and an employee. Figure 5.1-1 below illustrates these command relationships.

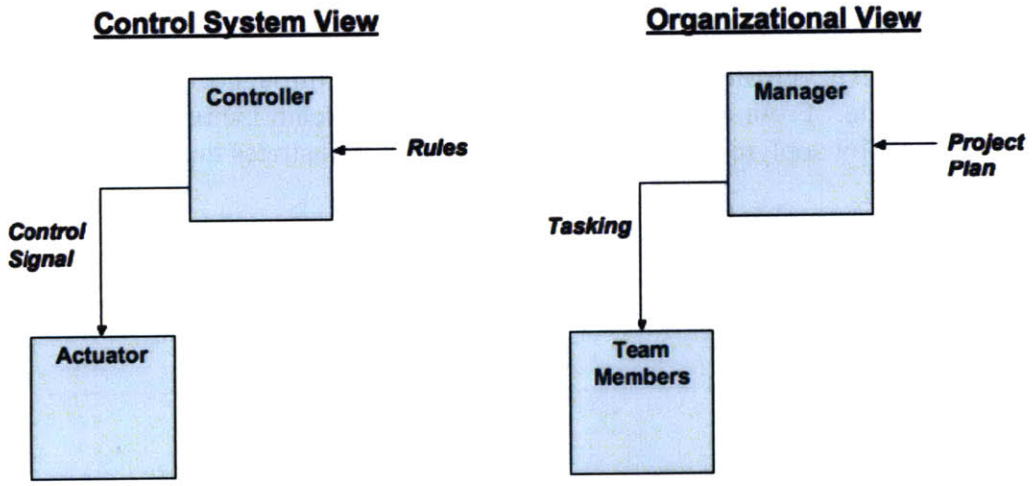

Figure 5.1-1 Command Relationships

Regardless of the organization's structure (e.g. function-oriented, product-oriented, or matrix) the organizational controllers and actuators can easily be identified using the RAM. Recall that the RAM not only captures reporting relationships, but also explicitly identifies accountable and responsible individuals. Accountable individuals are equivalent to controllers and responsible individuals are equivalent to actuators. Thus a hierarchical organization with high-level accountable managers overseeing lower level accountable managers is analogous to a hierarchical control system with high-level controllers overseeing lower level controllers.

The structural similarities between components in Figure 5.1-1 are apparent. But perhaps less apparent are similarities in the data passed on the interfaces between these components. Control signals issued by a controller are a command to an actuator to perform its function at a specified intensity. A control signal can be any information that can be converted by an actuator into a form capable of affecting a controlled process. In a similar fashion, tasking issued by a manager is a signal to a team to execute an assigned element of scope at a specified intensity. When coupled with EVM, tasking will typically define intensity in terms of planned EV accomplishment and AC spending rates.

One final command relationship between control systems and organizations can be found in the driving input data to the controller-manager component. Controllers are provided rules that define a goal for the control system. This goal can be to ensure that an aspect of a controlled process is maintained at a set point and kept within thresholds. Likewise, managers are provided project plans that define goals for their portion of the organization. These goals are typically to ensure that milestones are completed and kept within the allocated schedule and budget.

5.1.2 Controlled Process Relationships

In the previous section, controllers and actuators were explained to be analogous to managers and their team members. Another major relationship between control systems, organizations and project management is that of the controlled process. From a control systems perspective, actuators affect controlled processes through the manipulation of a controlled variable. From an organizational perspective, team members affect elements of project scope, by applying effort. Figure 5.1-2 below illustrates this controlled process relationship.

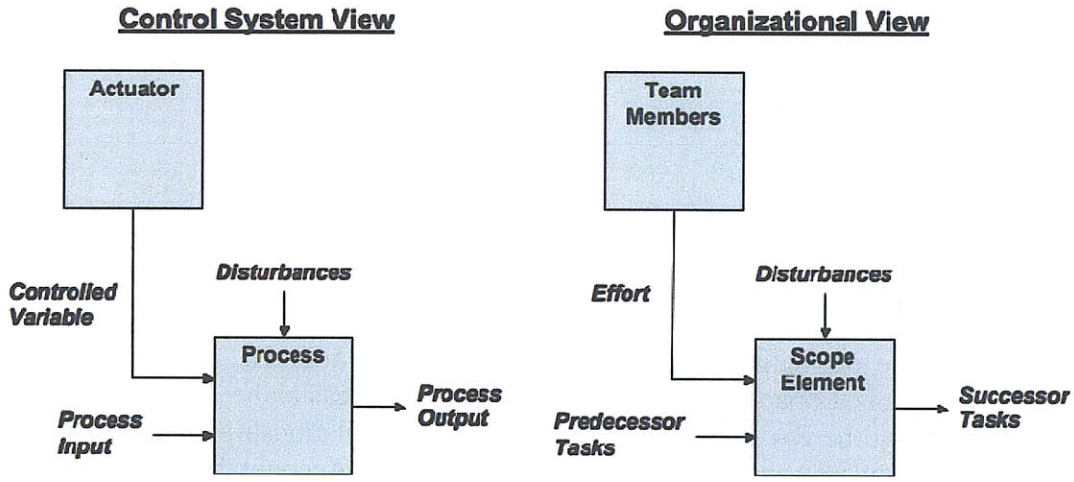

Figure 5.1-2 Controlled Process Relationships

Just as the controllers and actuators of a project can be identified using the RAM, a project's controlled processes can be identified using the WBS. The highest level of the WBS is the project itself. Regardless of the organizational scheme used to decompose the WBS, the scope elements at each level will represent the controlled processes of that level. From an organizational perspective, these controlled processes are aligned with the appropriate controller via the RAM. Recall that the RAM pairs accountable individuals with a scope element to control and pairs responsible individuals with a scope element to affect. Thus the concept of hierarchy applies equally to controlled processes as it does to their control systems.

One additional controlled process relationship to be aware of is that of the inputs and outputs of the controlled process independent of the presence of a control system. Every process has some resources it requires from an external entity, as well as results produced and provided to external entities. In project management terminology, controlled process inputs are equivalent to predecessor and successor scope elements respectively. These linkages are usually captured in a task dependency list or diagram.

5.1.3 Feedback Relationships

In control systems, a feedback loop is the controller's means of assessing its impact on the controlled process. The controlled process is monitored by a sensor, which in turn reports meaningful information back to the controller. Organizations also have feedback loops, where reporting tools are analogous to sensors. Reporting tools can take on many different forms, but just as the sensor monitors a controlled process, reporting tools monitor elements of scope. Managers make decisions based on the information provided by reporting tools, just as controllers make decisions based on the information provided by sensors. Figure 5.1-3 below illustrates this feedback relationship.

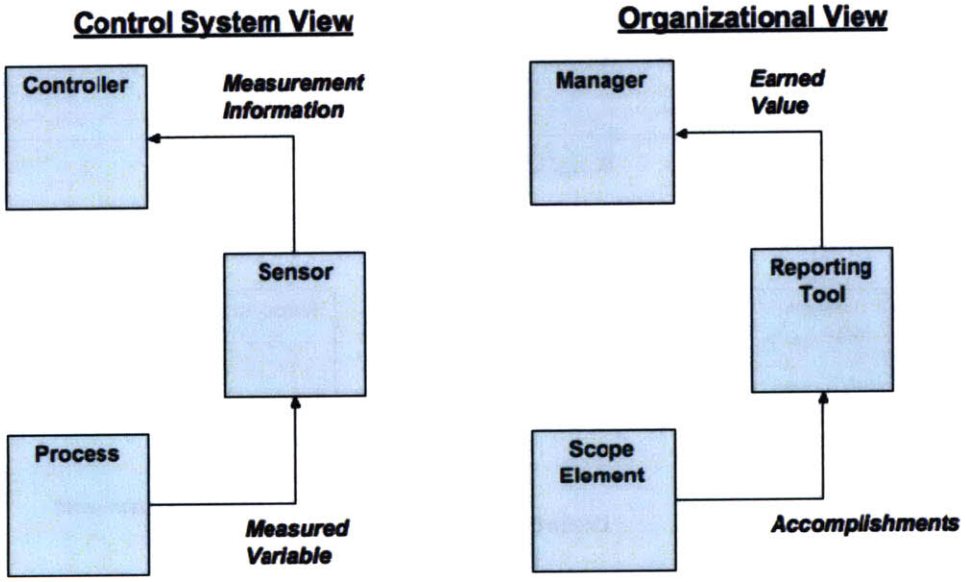

Figure 5.1-3 Feedback Relationships

There are two types of interfaces within the feedback loop of a control system: sensor-process and sensor-controller. In the sensor-process interface, the sensor monitors one or more controlled process variables. The thermostat example used throughout Section 4 showed that heat was the variable measured by the thermometer sensor. In the organizational context, reporting tools are used to monitor scope elements. Progress made on the scope element, in the form of milestones accomplished, is typically captured by the reporting tool. Another feedback example, from the organizational perspective, would be the timecard system that most companies use to capture hours expended by employees on specific tasks.

The second type of interface within a feedback loop is that of the sensor-controller. In the sensor-controller interface, the sensor converts the measured variable into meaningful information that can be acted upon by the controller. Returning to the thermostat example, the room heat was converted by the thermistor into a voltage. In the organizational context, reporting tools convert accomplished milestones into EV. This actual EV progress on a scope element can be analyzed by a manager and compared against that scope element's expected EV progress in the project plan.

5.1.4 The Organizational Control System

Portions of a generalized organizational control system have been presented throughout this section. Figure 5.1-4 below depicts the integrated form of this control system. Visualizing the comparisons between electro-mechanical and organizational control systems should be intuitive when one follows the flow of information through the command, controlled process and feedback loops. Managers task team members. Team members apply effort to project scope. Accomplishments against project scope are captured by reporting tools. Reporting tools convey relevant information back to managers. Managers provide new tasking based on observed progress. And so on.

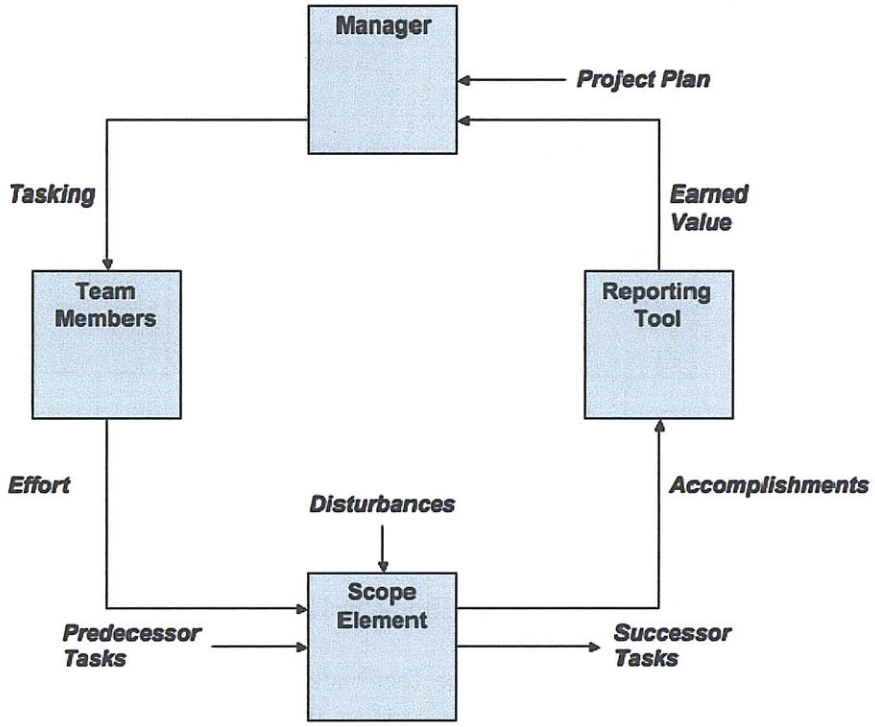

Figure 5.1-4 Organizational Control System

Hopefully, the many relationships between project management, organization structures and control systems are clear at this point. In order to develop a framework that will help project managers to avoid programmatic losses, we must treat the organization as a hierarchical control system that enforces constraints on elements of project scope. Let us now develop this framework by first specifying its goals and requirements.

5.2 Framework Specification

Once a contract is awarded, the first activity of a program management team is to develop a project plan, which will contain a definition of the elements of project scope, a list of dependencies between the elements, and an allocation of budget and schedule to each element. Let us refer to the result of this activity as the *project layer* of the program. Next the program management team will typically assemble an organization, based on a combination of product and domain knowledge, to execute the project plan. Let us refer to the result of this activity as the *organizational structure layer* of the program. In many cases it is common to assume that the program planning is complete and that work on the program can proceed, once these two layers have been defined. Unfortunately, a critical piece of the puzzle is still missing.

Programs are a composite of not just the project and organizational structure layers, but also an often ill-defined intermediary layer that we will refer to as the *control structure layer*. A portion of the control structure layer, the RAM, was discussed already in Section 3. But the RAM only serves as a tool to associate elements of the organizational structure layer with elements of the project layer (i.e. associating OBS controllers and actuators with WBS controlled processes). It does nothing to actually define the appropriate control system for a given scope element. To be clear, most project managers are good at defining the existence of a relationship between the organization and project, but poor at specifying how exactly an organizational controller should affect and receive feedback from the controlled scope. These omissions are depicted in Figure 5.2-1 below.

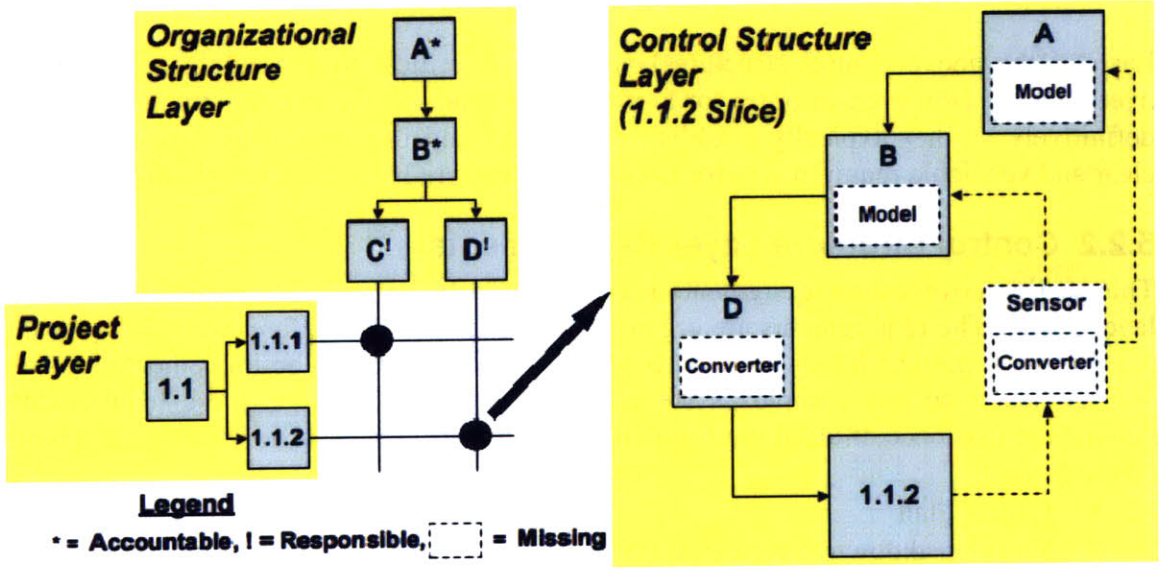

Figure 5.2-1 Typical Deficiencies in Control Structure Layer

The first two objectives of this thesis were to provide project managers with a mechanism to control risk within the scope of the work they oversee and to provide individual contributors with a mechanism to control risk within the scope of the work they execute. It is the author's belief that the project and organizational structure layers of the program framework are sufficiently well understood and that the lack of a specification for the control structure layer is the primary cause of risk that leads to program losses. With these objectives and assumptions in mind, the focus of this specification will be on the control structure layer of the program framework.

5.2.1 Control Structure Layer Goals

A goal is a desired end outcome for a system or process. The overall goal of most programs is to achieve the project scope within specified budget and schedule constraints. This program goal can be further decomposed into more specific goals for the project, organizational structure, and control structure layers. From the perspective of the control structure layer, there are five goals:

- G1: To enforce the budget and schedule constraints, as defined in the project plan, on all project scope elements
- G2: To monitor all project scope elements in a manner that will support analysis of the impact of control actions
- G3: To communicate current progress on all project scope elements to stakeholders on a regular basis
- G4: To communicate the forecasted end state of all project scope elements to stakeholders on a regular basis
- G5: To report predicted risks and actual losses, in terms of budget and schedule constraints, to stakeholders as rapidly as possible

Each of the above control structure layer goals is critical to a program achieving its overall goal. However, in order for a program to establish the control structure layer as definitively as they typically establish the project and organizational structure layers, clear and verifiable requirements for the control structure layer must be specified.

5.2.2 Control Structure Layer Requirements

This section covers the requirements necessary to achieve each of the control structure layer goals. The requirements are organized in a manner that captures traceability to the parent goal from which they were derived. They are intended to be implementation-free and applicable to any complex systems project. As a final note, these requirements assume the existence of all of the following program artifacts prior to initiating any work on the project:

- Project Plan
- Work breakdown structure (WBS)
- Organization Breakdown Structure (OBS)
- Responsibility Assignment Matrix (RAM)

5.2.2.1 Enforcing Constraints

The following requirements have been derived from goal G1 of the program framework's control structure layer. As previously stated, *the desired control structure must enforce the budget and schedule constraints, as defined in the project plan, on all project scope elements.* In order to achieve this goal, the proposed design must meet all of the following requirements:

- R1.1: The control structure ***shall*** include a cumulative history state model for each scope element.
    - The states within this model will be defined in terms of SV, SV%, CV, and CV%. Transitions between model states will be defined as a function of crossing SV, SV%, CP, and CP% thresholds defined at the inception of the program.
- R1.2: The control structure ***shall*** update the state model for a given scope element based on measurement information on that scope element.
    - The measurement information on scope elements will be defined in terms of EV and AC.
- R1.3: The control structure ***shall*** update the state model for a given scope element upon input from appropriate stakeholders of higher-level scope elements.
    - Appropriate stakeholders (i.e. parent controllers) associated with higher-level scope elements will be defined in the program RAM.
- R1.4: The control structure ***shall*** update the cumulative history state model for each active scope element at regular intervals.
    - These intervals will be defined at inception of the program.
- R1.5: The control structure ***shall*** initialize the model for each scope element based on the program PV schedule.
    - The PV schedule, coupled with the EV and AC observations, will be the basis for calculating state transition events.
- R1.6: The control structure ***shall*** take corrective actions to prevent scope elements from transitioning to a state where the budget and schedule constraints have been violated.
    - Budget and schedule constraint violations will be defined as a function of a given scope element's BAC.
- R1.7: The control structure ***shall*** only allow corrective actions for a scope element to be issued by the accountable manager of that scope element.
    - The association between accountable manager and scope element will be defined in the program RAM.
- R1.8: The control structure ***shall*** issue all corrective actions, in a manner that can be interpreted by responsible team members, as it applies to the project scope element with which they are associated.
    - The corrective actions will be defined in terms of SPI and CPI rates, and tied to defined items in the project plan.

5.2.2.2 Monitoring Scope

The following requirements have been derived from goal G2 of the program framework's control structure layer. As previously stated, *the desired control structure must monitor all project scope elements in a manner that will support analysis of the impact of control actions.* In order to achieve this goal, the proposed design must meet all of the following requirements:

- R2.1: The control structure ***shall*** measure and record the current and cumulative progress on each scope element, at regular intervals.
    - Scope element progress will be defined in terms of EV and AC.
    - These intervals will be defined at inception of the program.
- R2.2: The control structure ***shall*** provide the recorded current and cumulative scope element progress measurements to the accountable manager associated with that scope element, at regular intervals.
    - The relationship between scope element and accountable manager will be defined in the program RAM.
    - These intervals will be defined at inception of the program.

5.2.2.3 Communicating Progress

The following requirement has been derived from goal G3 of the program framework's control structure layer. As previously stated, *the desired control structure must communicate current progress on all project scope elements to stakeholders on a regular basis.* In order to achieve this goal, the proposed design must meet the following requirement:

- R3.1: The control structure ***shall*** provide the recorded current and cumulative scope element progress measurements for a given scope element to appropriate stakeholders in the hierarchy of that scope element, at regular intervals.
    - Scope element progress will be defined in terms of EV and AC.
    - The relationship between scope element and appropriate stakeholders (i.e. parent controllers) will be defined in the program RAM.
    - These intervals will be defined at inception of the program.
- R3.2: The control structure ***shall*** disseminate recorded current and cumulative scope element progress measurements for a given scope element to all accountable managers of directly related scope elements.
    - Scope element relationships (i.e. predecessor and successor tasks) are defined in the project plan's task dependency list.

5.2.2.4 Forecasting Completion

The following requirements have been derived from goal G4 of the program framework's control structure layer. As previously stated, *the desired control structure must communicate the forecasted end state of all project scope elements to stakeholders on a regular basis.* In order to achieve this goal, the proposed design must meet all of the following requirements:

- R4.1: The control structure ***shall*** calculate the estimated remaining cost to complete each active element of project scope, as a function of demonstrated cost performance.
    - This is the formal ETC formula based on either current period or averaged CPI.
- R4.2: The control structure ***shall*** provide the estimated final cost for a given scope element to appropriate stakeholders in the hierarchy of that scope element, at regular intervals.
    - The final cost of a scope element is the estimated or calculated remaining cost plus the cumulative AC.
    - The relationship between scope element and appropriate stakeholders (i.e. controllers and parent controllers) will be defined in the program RAM.
    - These intervals will be defined at inception of the program.

5.2.2.5 Reporting Risks and Losses

The following requirements have been derived from goal G5 of the program framework's control structure layer. As previously stated, *the desired control structure must report predicted risks and actual losses, in terms of budget and schedule constraints, to stakeholders as rapidly as possible.* In order to achieve this goal, the proposed design must meet all of the following requirements:

- R5.1: The control structure ***shall*** calculate the current period and cumulative variances in performance for each active element of project scope.
    - Performance variances will be defined in terms of SV, SV%, CV, and CV%.
- R5.2: The control structure ***shall*** calculate the predicted final variances in performance for each active element of project scope.
    - Final performance variances will be defined in terms of VAC, which can be calculated using the scope element's BAC and EAC.
- R5.3: The control structure ***shall*** provide the current period, cumulative and final predicted variances for a given active scope element to the appropriate stakeholders in the hierarchy of that scope element, at regular intervals.
    - The relationship between scope element and appropriate stakeholders (i.e. controllers and parent controllers) will be defined in the program RAM.
    - These intervals will be defined at inception of the program.
- R5.4 The control structure ***shall*** analyze the accumulated performance data from all active elements of project scope for evidence of potential risks and losses.
    - Scope element performance data will be aggregated in both WBS-aligned and OBS-aligned formats.
- R5.5 The control structure ***shall*** disseminate evidence of potential risks and losses to all accountable managers of directly related scope elements.
    - Evidence will clearly identify and communicate affected scope elements in the WBS and affected accountable managers and responsible team members from the OBS.

5.3 Framework Design

To reduce programmatic risk on complex systems projects, all of the requirements in Section 5.2.2 must be incorporated into the design of the program framework's control structure layer. Let us now allocate these requirements to elements of the control structure. Table 5.3-1 below depicts these allocations, along with their associated goals.

Table 5.3-1 Requirements Allocation

Control Structure Goal	Requirement	Component Allocation
G1 - Enforcing Constraints	R1.1	Managers
	R1.2	Managers
	R1.3	Managers
	R1.4	Managers
	R1.5	Managers
	R1.6	Managers, Team Members
	R1.7	Managers, Team Members
	R1.8	Managers, Team Members
G2 – Monitoring Scope	R2.1	Reporting Tools
	R2.2	Reporting Tools
G3 – Communicating Progress	R3.1	Reporting Tools
	R3.2	Managers
G4 – Forecasting Completion	R4.1	Reporting Tools
	R4.2	Reporting Tools
G5 – Reporting Risks and Losses	R5.1	Reporting Tools
	R5.2	Reporting Tools
	R5.3	Reporting Tools
	R5.4	Managers
	R5.5	Managers

5.3.1 Manager Design

Controllers are the decision-makers of a control system. The signals they issue alert actuators to perform their function at a specified intensity. In this framework, managers make decisions based upon feedback on the scope element under their control and issue *Tasking* to members of their team. The RAM identifies all accountable managers on the program beginning with the program manager (i.e. individual accountable for the highest WBS element) and ending with the program's team leads (i.e. set of individuals accountable for the lowest level of WBS elements). This design is a generic template for any manager component in the control structure, but note that only the lowest level managers are able to issue direct *Tasking* to team members.

The following requirements are fully or partially allocated to each manager component within the control structure layer: R1.1, R1.2, R1.3, R1.4, R1.5, R1.6, R1.7, R1.8, R3.2, R5.4, and R5.5.

5.3.1.1 Manager Initialization

At the inception of the program, all manager components within the control structure will initialize the data items shown in Table 5.3-2 below.

Table 5.3-2 Manager Initial Settings

Data Item	Source	Description
Scope Element ID	RAM	Unique ID number of the scope element to be controlled by the manager
Team Members	RAM	Responsible individual(s) for the scope element; only relevant for scope elements at lowest level of WBS
Command Chain	RAM	Set of accountable individuals for all parent scope elements of this manager's scope element; not relevant for highest level scope element (i.e. program manager is accountable, with no one to report to)
Measurement Interval	Project Plan	Time period between successive measurements of progress on the controlled scope element
PV Plan	Project Plan	Set of milestone names, dates, and associated PV for the scope element; these are defined for the lowest elements of the WBS, and then aggregated for parent WBS elements
BAC	Project Plan (Derived)	Sum of all PV for all milestones of the scope element
Cumulative State Model	N/A	This model is the manager's representation of the scope element's current state; the Cumulative State Model has a starting state of *Unitiated.*
Risk Thresholds	Project Plan (Derived)	Set of thresholds to be used as transition events within the *Cumulative State Model*; these include: *RiskSV, RiskSV%, RiskCV,* and *RiskCV%*

5.3.1.2 State Model Design

A manager's ability to effectively control a scope element is dependent on being able to analyze feedback on that scope element and to issue appropriate commands to their team members. In this framework, manager components are provided with a model that represents the best information available on the state of the scope element. The *Cumulative State Model* captures the state of the scope element as of the most recent reporting period. The model incorporates scope, schedule, and budget performance data as well as predictive data for the end state of the scope element. It is through this model that the manager is able to essentially convert feedback from the reporting tool into meaningful *Tasking* to team members.

5.3.1.2.1 State Definitions

The cumulative state model is comprised of six states that together represent any possible scenario for an active or inactive scope element. These states are a function of the scope element parameters *Cumulative EV, Cumulative PV, Cumulative AC, Cumulative SV, Cumulative SV%, Cumulative CV,* and *Cumulative CV%.* The states are defined as follows:

- Unitiated State – This state indicates that the scope element is inactive, has not had progress towards its milestones, and has not yet consumed any resources.
- Nominal – This state indicates that the scope element is active and progressing at or ahead of both schedule and budget. Scope is incomplete.
- Warning - This state indicates that the scope element is active, is behind either budget or schedule, but still ahead of all risk thresholds. Scope is incomplete.
- Risk – This state indicates that the scope element is active and has tripped either a budget or schedule risk threshold, but has not yet violated a budget or schedule constraint. Scope is incomplete.
- Loss – This state indicates that the scope element is inactive and has violated either a budget or schedule constraint.
- Completed – This state indicates that the scope element is inactive, that all milestones have been accomplished, and that no constraints were violated.

5.3.1.2.2 Possible State Transitions

While it is possible for a scope element to eventually reach any of the defined states, it is not always possible to transition directly between two states. All cumulative state models on the project are initialized in the Unitiated state, with the defining parameters all set to zero. State models transition from Unitiated to Nominal, at the beginning of planned work. On the rare occasion when unauthorized work or spending has occurred prior to the planned start date, state models may transition immediately from Unitiated to the Warning, Risk, or Loss states. Table 5.3-3 below identifies all possible state transitions, with the aforementioned special transitions indicated with a "Y' ".

Table 5.3-3 Possible State Transitions

		To State [Y]					
	States	**Unitiated**	**Nominal**	**Warning**	**Risk**	**Loss**	**Completed**
From State [X]	**Unitiated**	Y	Y	Y'	Y'	Y'	
	Nominal		Y	Y	Y	Y	Y
	Warning		Y	Y	Y	Y	Y
	Risk		Y	Y	Y	Y	Y
	Loss					Y	
	Completed						Y

Manager components transition their state models based on transition event rules and information received from reporting tool components. These transition events are explained, in the next section, as part of the manager's Update Model function.

5.3.1.3 Update Model Function

The manager *Update Model Function* is the means by which all manager components within the control structure are able to calibrate their understanding of the current state of the scope elements under their control. All state models begin in the *Unitiated* state and these models are updated when feedback is received from the scope element's reporting tool.

- Initiate function upon receipt of external *Scope Element Progress Report.*
- Replace current model values for *Cumulative EV, Cumulative PV, Cumulative AC, Cumulative SV, Cumulative SV%, Cumulative CV,* and *Cumulative CV%* with the corresponding values from the received *Scope Element Progress Report.*
    - These updated values are the basis for state transition events.
- IF *(Cumulative EV = BAC)* AND *(Cumulative AC <= BAC),* THEN...
    - Transition the model into the *Completed* state.
    - Issue internal signal *Cease Operations.*
- IF *(Cumulative PV = BAC)* AND *(Cumulative EV < BAC),* THEN...
    - Transition the model into the *Loss* state.
    - Issue internal signal *Cease Operations.*
- IF *(Cumulative AC >= BAC),* THEN...
    - Transition the model into the *Loss* state.
    - Issue internal signal *Cease Operations.*
- IF *(Cumulative PV < BAC)* AND *(Cumulative EV < BAC)* AND *(Cumulative AC < BAC)* AND *(Cumulative SV >= 0)* AND *(Cumulative CV >= 0),* THEN...
    - Transition the model into the *Nominal* state.
    - Issue internal signal *Update Complete.*
- IF *(Cumulative PV < BAC)* AND *(Cumulative EV < BAC)* AND *(Cumulative AC < BAC)* AND *((0 > Cumulative SV >= RiskSV)* OR *(0 > Cumulative SV% >= RiskSV%)* OR *(0 > Cumulative CV >= RiskCV)* OR *(0 > Cumulative CV% >= RiskCV%)),* THEN...
    - Transition the model into the *Warning* state.
    - Issue internal signal *Update Complete.*
- IF *(Cumulative PV < BAC)* AND *(Cumulative EV < BAC)* AND *(Cumulative AC < BAC)* AND *((RiskSV > Cumulative SV)* OR *(RiskSV% > Cumulative SV%)* OR *(RiskCV > Cumulative CV)* OR *(RiskCV% > Cumulative CV%)),* THEN...
    - Transition the model into the *Risk* state.
    - Issue internal signal *Update Complete.*

5.3.1.4 Command Function

The manager *Command Function* is the means by which all manager components within the control structure alert subordinates as to which milestones to work on, as well as the necessary schedule and cost efficiency required to maintain the project plan.

- Initiate function upon receipt of internal signal *Update Complete.*
- For all planned milestones that are incomplete as of the current period as well as any planned milestones in the upcoming period...
    - Set $Commanded\ SPI = \frac{BAC - CumulativeEV}{BAC - CumulativePV}$

- Set *Commanded CPI* = $\frac{BAC - CumulativeEV}{BAC - CumulativeAC}$

- Generate *Tasking*.
  - The *Tasking* signal will contain *Manager, Team Members, Scope Element ID, and Tactical Plans.*
  - *Tactical Plans* are a set of one or more *Milestones* from the PV plan along with *Commanded SPI* and *Commanded CPI* for each.
- Issue *Tasking* to *Team Members.*

As the reader may have recognized, the designed values for *Commanded SPI* and *Commanded CPI* represent a very simplistic P-Controller. In Section 6.2.3, we will consider the impact of setting *Tasking* based on other types of controllers.

5.3.1.5 Verify Function

The manager *Verify Function* checks all received control signals for validity prior to acting upon that control signal.

- Initiate function upon receipt of external signal *Tasking.*
  - The *Tasking* signal will contain *Manager, Scope Element ID, and Tactical Plans.*
  - *Tactical Plans* are a set of one or more *Milestones* from the PV plan along with *Commanded SPI* and *Commanded CPI* for each.
- Check *Manager*
  - This check will ensure that the *Manager* that issued the *Tasking* signal is in fact one of the accountable individuals identified in *Command Chain* (i.e. authorized to direct the manager of this scope element).
- Check *Scope Element ID*
  - This check will ensure that the *Scope Element ID* identified in the received *Tasking* signal matches the *Scope Element ID* (i.e. accountability) of the *Manager* who received the *Tasking* signal.
- Issue internal signal *Verification Complete.*

5.3.1.6 Flowdown Function

The manager *Flowdown Function* is the means by which manager components within the control structure alert subordinates to discontinue all work on a scope element and also the means by which a parent manager can command a subordinate manager to use a custom *Commanded SPI* or *Commanded CPI* (i.e. circumvent that manager's state model).

- Initiate function upon receipt of internal signal *Cease Operations* or internal signal *Verification Complete.*
- IF (*Cease Operations*), THEN...
  - Generate *Tasking*.
    - The *Tasking* signal will contain *Manager, Team Members, Scope Element ID, and Tactical Plans.*
    - *Tactical Plans* are a set of one or more *Milestones* from the PV plan, with both *Commanded SPI* and *Commanded CPI* set to 0.

- Issue *Tasking* to *Team Members.*

- IF (*Verification Complete*), THEN...
  - Generate *Tasking.*
    - The *Tasking* signal will contain *Manager, Team Members, Scope Element ID, and Tactical Plans.*
    - *Tactical Plans* are a set of one or more *Milestones* from the PV plan along with *Commanded SPI* and *Commanded CPI* for each.
    - *Issue Tasking* to *Team Members.*

5.3.2 Team Member Design

Actuators are the means by which a control system is able to affect change in a controlled process. In this framework, team members apply effort towards accomplishing milestones for a given element of project scope. A perfect team would be able to self-manage and act as necessary to implement the project plan. But no team is perfect and factors such as skill level, domain experience, and work efficiency lead to deviations between commanded effort and actual effort. These same factors also impact the expenditure of resources such as time and money. As would be expected, the selection of the correct team members is equally as important to organizations as the selection of the proper actuator is to the success of a control system.

The following requirements are fully or partially allocated to each team member component within the control structure layer: R1.6, R1.7, and R1.8.

5.3.2.1 Team Member Initialization

At the inception of the program, all team member components within the control structure will initialize the data items shown in Table 5.3-4 below.

Table 5.3-4 Team Member Initial Settings

Data Item	Source	Description
Scope Element ID	RAM	Unique ID number of the scope element for which the team member is responsible
Manager	RAM	Accountable individual for the scope element and source of team member tasking
PV Plan	Project Plan	Set of milestone names, dates, and associated PV for the scope element; these are defined for the lowest elements of the WBS and then aggregated for parent WBS elements

5.3.2.2 Verify Function

The team member *Verify Function* checks all received control signals for validity prior to acting upon that control signal.

- Initiate function upon receipt of external signal *Tasking.*
  - The *Tasking* signal will contain *Manager, Team Members, Scope Element ID, and Tactical Plans.*

- *Tactical Plans* are a set of one or more *Milestones* from the PV plan along with *Commanded SPI* and *Commanded CPI* for each.

- Check *Manager.*
  - This check will ensure that the *Manager* that issued the *Tasking* signal matches the *Manager* to whom the *Team Members* report.
- Check *Team Members.*
  - This check will ensure that the *Team Members* that received the *Tasking* signal are in fact the intended recipients.
- Check *Scope Element ID.*
  - This check will ensure that the *Scope Element ID* identified in the received *Tasking* signal matches the *Scope Element ID* (i.e. responsibility) of the *Team Members* who received the *Tasking* signal.
- Issue internal signal *Verification Complete.*

5.3.2.3 Act Function

The team member *Act Function* is the actual expenditure of resources that occurs in the process of accomplishing the *Milestones* of the scope element.

- Initiate function upon receipt of internal signal *Verification Complete.*
- Execute *Milestones* at the *Commanded SPI* and *Commanded CPI.*
  - Note that if *Commanded SPI* and *Commanded CPI* are set to 0, work must terminate immediately.
- Issue *Actual EV* and *Actual AC* to *Scope Element ID.*

5.3.3 Reporting Tool Design

For any feedback-based control system, the sensor component is the controller's primary means of calibrating its internal model and assessing its impact on the controlled process. In this framework, reporting tools are the feedback mechanism for a manager to calibrate their state models and to assess the impact their tasking has had on a scope element. Reporting tool components can be implemented as software or as dedicated members of the organization. There can be a single or multiple instantiations of the reporting tool component within a program, so long as the implementation supports the monitoring and maintenance of each unique scope element.

The following requirements are fully or partially allocated to each reporting tool component within the control structure layer: R2.1, R2.2, R3.1, R4.1, R4.2, R5.1, R5.2, and R5.3.

5.3.3.1 Reporting Tool Initialization

At the inception of the program, all reporting tool components within the control structure will initialize the data items shown in Table 5.3-5 below.

Table 5.3-5 Reporting Tool Initial Settings

Data Item	Source	Description
Scope Element ID	RAM	Unique ID number of the scope element to be monitored by the reporting tool
Manager	RAM	Accountable individual for the scope element
Team Members	RAM	Responsible individual(s) for the scope element; only relevant for scope elements at lowest level of WBS
Command Chain	RAM	Set of accountable individuals for all parent scope elements of this reporting tool's scope element; not relevant for highest level scope element (i.e. program manager is accountable, with no one to report to)
Measurement Interval	Project Plan	Time period between successive measurements of progress on the monitored scope element
PV Plan	Project Plan	Set of milestone names, dates and associated PV for the scope element; these are defined for the lowest elements of the WBS and then aggregated for parent WBS elements
BAC	Project Plan (Derived)	Sum of all PV for all milestones of the scope element

5.3.3.2 Measurement Function

The reporting tool *Measurement Function* monitors the progress on a scope element. The steps in this function apply only to the scope element identified by a given reporting tool's *Scope Element ID.*

- Initiate function upon first milestone start date and then again once every *Measurement Interval* thereafter.
- Measure *Current Period EV.*
  - *Current Period EV* is measured for each milestone identified in *PV Plan. Current Period EV* should be set to 0 for incomplete milestones and be set to the full *PV*, as identified in the *PV Plan*, for any complete milestones.
- Measure *Current Period AC.*
  - *Current Period AC* is measured for *Manager* and *Team Members.*
- Record *Current Period EV* and *Current Period AC.*
- Issue internal signal *Measurement Complete.*

5.3.3.3 Conversion Function

The reporting tool *Conversion Function* performs all the calculations necessary to convert measured scope element progress data into useful information for the *Manager* and *Command Chain.* The steps in this function apply only to the scope element identified by a given reporting tool's *Scope Element ID.*

- Initiate function upon receipt of internal signal *Measurement Complete.*
- Calculate *Current Period PV.*

- *Current Period PV* is the sum of all *PV* of milestones to be completed in the current period.

- Calculate *Current Period SV.*
  - $Current\ Period\ SV = CurrentPeriodEV - CurrentPeriodPV$
- Calculate *Current Period SV%.*
  - $Current\ Period\ SV\% = \frac{CurrentPeriodSV}{CurrentPeriodPV}$
- Calculate *Current Period SPI.*
  - $Current\ Period\ SPI = \frac{CurrentPeriodEV}{CurrentPeriodPV}$
- Calculate *Current Period CV.*
  - $Current\ Period\ CV = CurrentPeriodEV - CurrentPeriodAC$
- Calculate *Current Period CV%.*
  - $Current\ Period\ CV\% = \frac{CurrentPeriodCV}{CurrentPeriodEV}$
- Calculate *Current Period CPI.*
  - $Current\ Period\ CPI = \frac{CurrentPeriodEV}{CurrentPeriodAC}$
- Calculate *Cumulative EV.*
  - $Cumulative\ EV = \sum CurrentPeriodEV$
- Calculate *Cumulative PV.*
  - $Cumulative\ PV = \sum CurrentPeriodPV$
- Calculate *Cumulative AC.*
  - $Cumulative\ AC = \sum CurrentPeriodAC$
- Calculate *Cumulative SV.*
  - $Cumulative\ SV = CumulativeEV - CumulativePV$
- Calculate *Cumulative SV%.*
  - $Cumulative\ SV\% = \frac{CumulativeSV}{CumulativePV}$
- Calculate *Cumulative SPI.*
  - $Cumulative\ SPI = \frac{CumulativeEV}{CumulativePV}$
- Calculate *Cumulative CV.*
  - $Cumulative\ CV = CumulativeEV - CumulativeAC$
- Calculate *Cumulative CV%.*
  - $Cumulative\ CV\% = \frac{CumulativeCV}{CumulativeEV}.$
- Calculate *Cumulative CPI.*
  - $Cumulative\ CPI = \frac{CumulativeEV}{CumulativeAC}$
- Calculate *Small Window ETC.*
  - $Small\ Window\ ETC = \frac{BAC - CumulativeEV}{CurrentPeriodCPI}$

- Calculate *Small Window EAC.*
  - $Small\ Window\ EAC = SmallWindowETC + CumulativeAC$
- Calculate *Small Window VAC.*
  - $Small\ Window\ VAC = BAC - SmallWindowEAC$
- Calculate *Full Window ETC.*
  - $Full\ Window\ ETC = \frac{BAC - CumulativeEV}{CumulativeCPI}$
- Calculate *Full Window EAC.*
  - $Full\ Window\ EAC = FullWindowETC + CumulativeAC$
- Calculate *Full Window VAC.*
  - $Full\ Window\ VAC = BAC - FullWindowEAC$
- Record the results of all above calculations.
- Issue internal signal *Conversion Complete.*

5.3.3.4 Reporting Function

The reporting tool *Reporting Function* compiles the raw data from the *Measurement Function* and the calculated data from the *Conversion Function* into a *Scope Element Progress Report.* This report is sent to the scope element's *Manager* and *Command Chain.* The steps in this function apply only to the scope element identified by a given reporting tool's *Scope Element ID.*

- Initiate function upon receipt of internal signal *Conversion Complete.*
- Generate *Scope Element Progress Report.*
  - Report contents are: *Scope Element ID, BAC, Current Period EV, Current Period AC, Current Period PV, Current Period SV, Current Period SPI, Current Period CV, Current Period CPI, Cumulative SV, Cumulative SPI, Cumulative CV, Cumulative CPI, Cumulative EV, Cumulative AC, Small Window ETC, Small Window EAC, Small Window VAC, Full Window ETC, Full Window EAC, Full Window VAC*
- Issue *Scope Element Progress Report* to *Manager.*
- Issue Scope Element Progress Report to Command Chain.

5.4 Closing

This section has explained the relationships between a program's project layer, organizational layer, and control structure layer. This section has also clearly stated a specification and design for a robust structure that will address the lack of control found on most programs. The integrated form of this control structure is depicted in Figure 5.4-1 below. We will next examine why program losses occur in a typical organization and illustrate how the proposed framework will alleviate these issues.

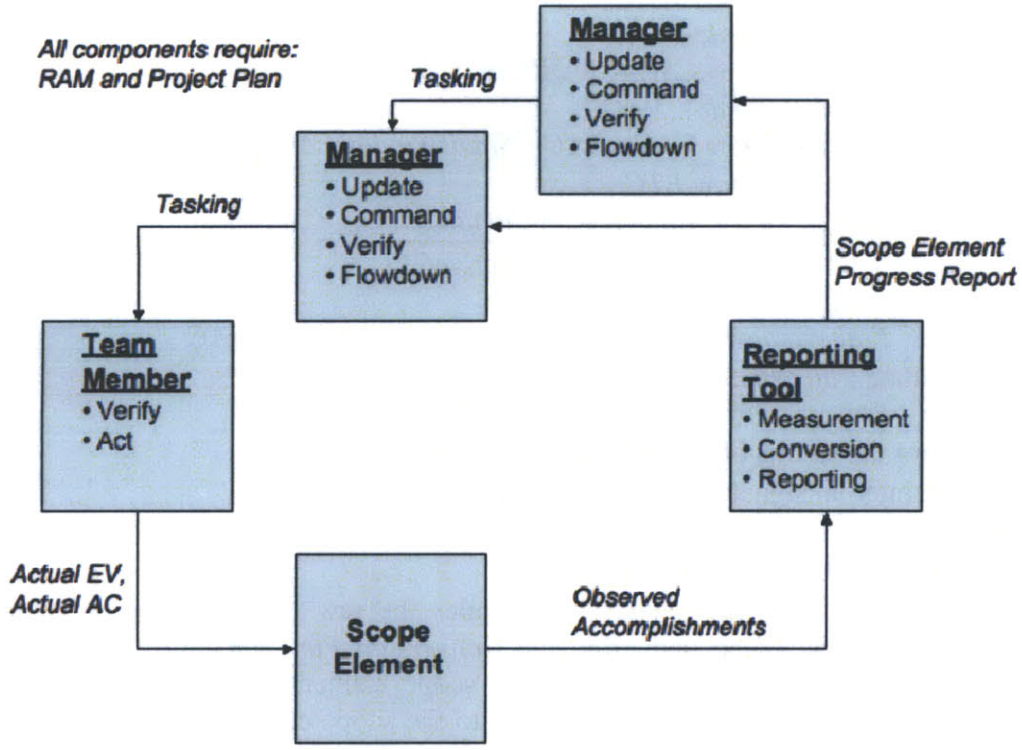

Figure 5.4-1 Proposed Control Structure Design

6 Case Study

By providing risk-control mechanisms for both project managers and their team members, the framework proposed in Section 5 has addressed two of the objectives of this thesis. However, two objectives remain: to clearly demonstrate how flaws in an organizational control structure can facilitate program losses, and to clearly demonstrate how a well designed organizational control structure can mitigate or even prevent these losses.

In this section, we will present the organizational control structure of an actual complex system program. Our focus will be on the static and dynamic properties of the communication network that links teams within the organization to each other. We will highlight areas within the existing control structure that have the potential to facilitate program losses, by drawing comparisons to the sources of control system error defined in Section 4.3. Based on the control structure layer specification and design defined in Section 5, we will recommend improvements to the existing control structure for each of these areas of concern.

6.1 Example Program

This thesis opened by discussing a history of budget and schedule overruns on major Department of Defense (DoD) programs. Therefore, it seems only fitting that we should use a DoD program as the thesis case study. The author does not intend to imply that this particular program has contributed in any way to the overall cumulative overruns of the DoD's major program portfolio. All information presented in this case study is public knowledge and most of the technical details about the program scope (i.e. the project layer) have been intentionally omitted, as they are irrelevant to this thesis. All names have been changed, and the team responsibility is captured only as it relates to the organizational structure and control structure layers.

The ALPHA program is responsible for the design, development, testing, deployment, and maintenance of the ALPHA Weapon System (AWS). The AWS is an element within the Ballistic Missile Defense System that protects the United States, its deployed forces, and its allies. A fully functional AWS is comprised of a phased-array radar and a battery of interceptor missiles. During operations, the AWS radar is used to search, detect, track, and discriminate ballistic missile threats. If authorized by a battlefield commander, a threat can be intercepted and destroyed by the AWS missile. During an engagement, the AWS radar is utilized to guide the AWS missile to the threat as well as to verify the success or failure of the intercept. AWS is just one complex system within a larger network of complex systems. The elements that comprise the Ballistic Missile Defense System are linked together and coordinated by a central Command and Control element.

The ALPHA organization is comprised of approximately 700 engineers and support staff. This organization is structured as a matrix similar to the one described in Section 3.1.3. The exact structure of this organization will be explained shortly, but for now it is sufficient to note that individuals can belong to a Cross-Product Team (i.e. function-oriented), an Integrated-Product Team (i.e. product-oriented), or both. Scope within the ALPHA program is structured at the highest-level by distinct government contracts. These contracts are then decomposed into their constituent scope elements and assigned to an accountable lead. At any given time, Cross-Product Team (CPT) members can be executing scope elements on multiple contracts and product lines, while Integrated-Product Team (IPT) members focus on completing the scope on a single contract.

Let us now examine the organizational structure and control structure layers of the ALPHA program.

6.1.1 ALPHA Organizational Structure Layer

The ALPHA program's organizational structure layer is a matrix of four IPTs and four CPTs. The leads of these IPTs and CPTs are direct reports to the ALPHA program director. This director serves as ALPHA's corporate point of contact, sets strategic goals for the organization, and arbitrates disputes between the IPT and CPT leads. The OBS in Figure 6.1-1 below depicts the reporting relationships within ALPHA.

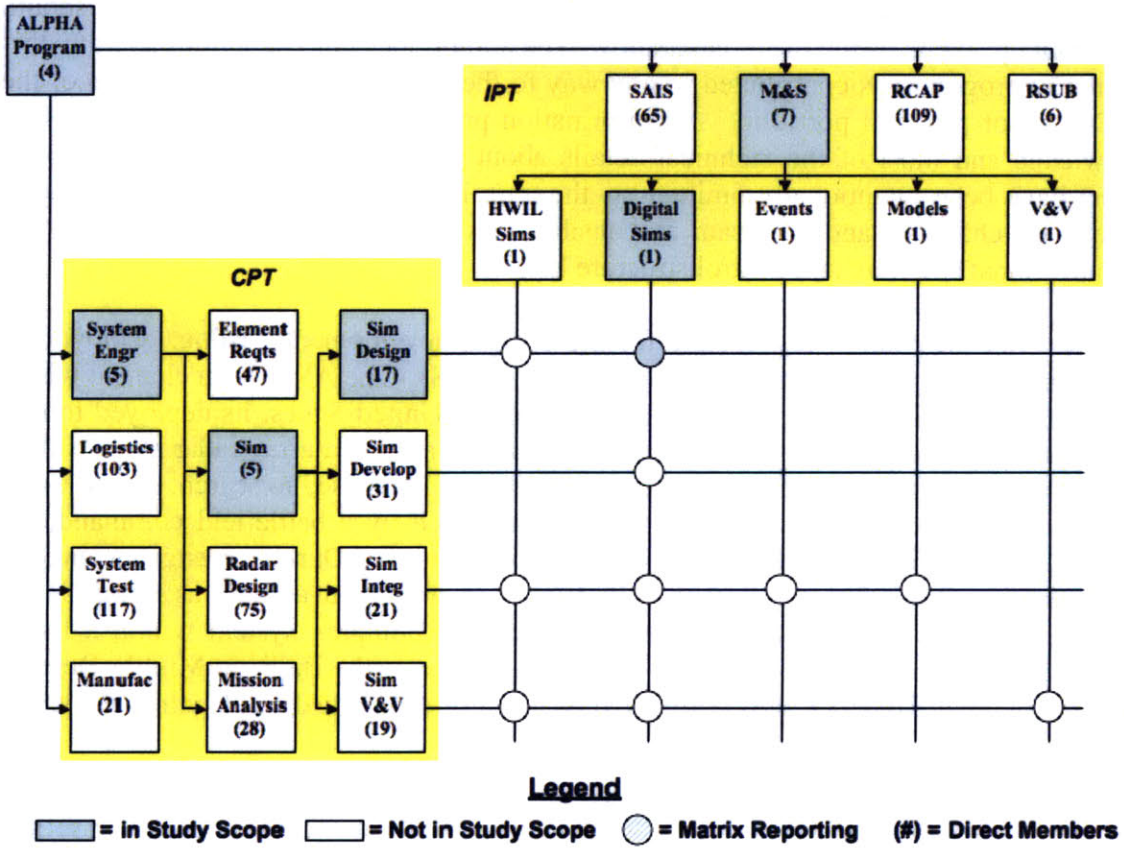

Figure 6.1-1 ALPHA Program Engineering OBS (Partial)

Figure 6.1-1 is only a partial decomposition of the ALPHA program's organizational structure. The control and communications analysis in this case study will focus on the items in blue, but the same principles apply equally to all of the depicted teams. Each box in the figure contains the name of a team as well as the number of individuals who are direct, as opposed to reporting, members of the team. The circles within the grid represent matrix-reporting relationships, temporary situations where CPT members have been assigned to execute scope and are held accountable to IPT members. Since a matrix is actually both a functional and product-oriented organization existing simultaneously, let us go into some detail as to the nature of the ALPHA CPTs and IPTs.

Cross Product Teams are the program's domain knowledge experts. These teams are not tied to any one contract, but instead support all contracts that require their skill set. The ALPHA organization is comprised of four CPTs: Systems Engineering and Integration (SEI), Logistics and Specialty Engineering (LSE), Radar System Test (RST), and Manufacturing (MFT). Each of these CPTs is managed by a single lead that reports only to the program director and is supported by several staff members. A brief description of each CPT is as follows:

- SEI: This CPT has 248 members distributed across the domains of Element Requirements, Simulations, Radar Design and Mission Planning. The Simulations sub-CPT is the focus of our study; it is comprised of specialists in the areas of Simulation Design, Simulation Development, Simulation Integration, and Simulation Verification and Validation.
- LSE: This CPT has 103 members distributed across the domains of Logistics Engineering, Specialty Engineering, Training and Technical Manual Development, and Systems Sustainment.
- RST: This CPT has 117 members distributed across the domains of Flight Test, Ground Test, Radar System Verification, Specialty Testing, and Mission Execution.
- MFT: This CPT has 21 members distributed across the domains of Antenna Hardware, Electronics Hardware, Cooling Hardware, Power Systems Hardware, and Sustainment.

Integrated Product Teams are the program's customer representatives. These teams are each focused on only one contract and more specifically, the deliverable product lines within that contract. The ALPHA organization is comprised of four IPTs: Systems Architecture Interface and Support (SAIS), Models and Simulations (MS), Radar Capabilities (RCAP), and Radar Sub-Systems (RSUB). Each of these IPTs is managed by a single lead that reports only to the program director, and is supported by several staff members. A brief description of each CPT is as follows:

- SAIS: This IPT has 65 members who together oversee products spanning Radar System Architecture, Future Systems, and Logistics. SAIS is abnormally large for an IPT because it has two permanently assigned teams in the domains of Weapon System Integration, and Fire Control and Communications.
- MS: This IPT has 12 members who together oversee products spanning Hardware-in-the-Loop Simulators, Digital Simulators, Simulation Events, Models, and Verification and Validation.

- RCAP: This IPT has 109 members who together oversee products spanning the Fielded Capabilities (i.e. all active software builds). RCAP is abnormally large for an IPT because it has a permanently assigned team in the domain of Software Development.
- RSUB: This IPT has 6 members who together oversee products spanning the Antenna Hardware, Electronics Hardware, Cooling Hardware, and Power Systems Hardware.

In a matrix organization like ALPHA, IPTs assemble their teams almost entirely out of CPT members. During the planning stage of each new contract, CPT leads partner with IPT leads to provide staffing from the appropriate sub-CPTs. At this point, sub-CPT leads select accountable task managers (i.e. control account managers) to be loaned to a sub-IPT lead for the duration of the relevant scope. These control account managers (CAMs) have complete control over forming their teams and sub-teams by drawing from other members of the CPT. At the completion of the project scope, the CAM teams are dissolved and receive new IPT assignments from their sub-CPT Lead. As previously stated, the SAIS and RCAP IPTs have identified a permanent role for their Weapon System Integration, Fire Control and Communications, and Software Development CAM teams. The individuals on these teams will never be reassigned to another IPT.

6.1.2 ALPHA Control Structure Layer

The ALPHA program's control structure layer links the work defined in the project layer to managers and team members identified in the organizational structure layer. Through the RAM, contract scope elements from the WBS are assigned to accountable and responsible IPT and CPT members. ALPHA's control structure layer also contains a communications network that monitors the progress and efficiency of contract scope execution. Let us now examine a portion of ALPHA's RAM and the communications network with which it is associated.

6.1.2.1 ALPHA RAM

Within ALPHA's RAM, as in any well-defined RAM, is a clear mapping of scope elements to accountable and responsible individuals. Recall from Section 3.2.1 that an *accountable* individual is one who the project manager has entrusted to monitor and deliver a scope element to the customer, and that a *responsible* individual is one who expends effort to complete that scope element. There can be only one accountable individual per scope element, but the same accountable individual can oversee multiple scope elements simultaneously. The same relationship is true for responsible individuals and the execution of scope elements. Below, Figure 6.1-2 depicts a slice of the ALPHA WBS along with the accountable and responsible individuals from the OBS. The figure does not depict the full extent of accountability and responsibility for these individuals, but it does nicely illustrate a single thread through the ALPHA control structure.

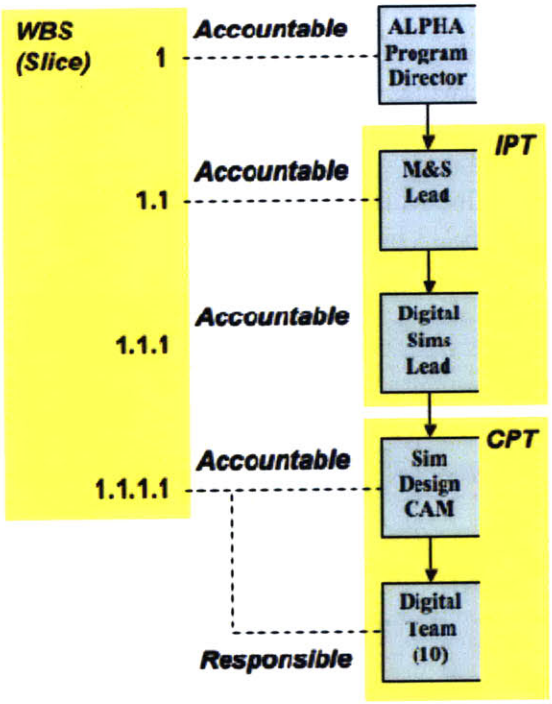

Figure 6.1-2 ALPHA RAM Excerpt

Accountability at the lowest level of contract scope is typically assigned to CPT CAMs; while accountability at higher levels is almost exclusively assigned to IPT leads. With only a few exceptions, responsibility for the lowest level elements of ALPHA contract scope is assigned to CPT members. The ALPHA program director is of course ultimately accountable for the successful completion and delivery of all contract scope. Noticeably absent from this RAM are members of the CPT leadership. This is due to the fact that in a matrix organization, CPT leads are responsible for growing domain knowledge and for fostering inter-product line communication. Aside from the CAMs, CPT members are not accountable for scope execution.

6.1.2.2 ALPHA Communications Network

Any network can be defined in terms of its nodes and links. In the case of the ALPHA program, the communications network is a set of manager and team member nodes that are linked together by the reporting relationships defined in the OBS. Formally, technical information is shared on a weekly basis when all members of a level meet together with their common lead (e.g. all CAMs working on 1.1.1.x scope meet weekly with the Digital Simulations Lead). Of course, managers and team members working on related scope elements are in much more frequent, informal contact.

Along the formal communication channels, higher-level managers provide subordinates with tasking; this pattern begins with the ALPHA program director, goes through the IPT leadership, goes down through the CAMs, and then ultimately reaches the relevant team members for a scope element. On a monthly basis, progress is reported at the lowest level of scope and is then aggregated and analyzed for higher-level scope elements.

Aspects of an organizational control system can be found embedded throughout the ALPHA communications network. All of the leads from the program director down through the CAMs represent controller components, while team members at the lowest level of scope represent actuator components. Progress reporting tools and individuals who specialize in Earned Value Management represent sensor components. This control system is depicted in Figure 6.1-3 below.

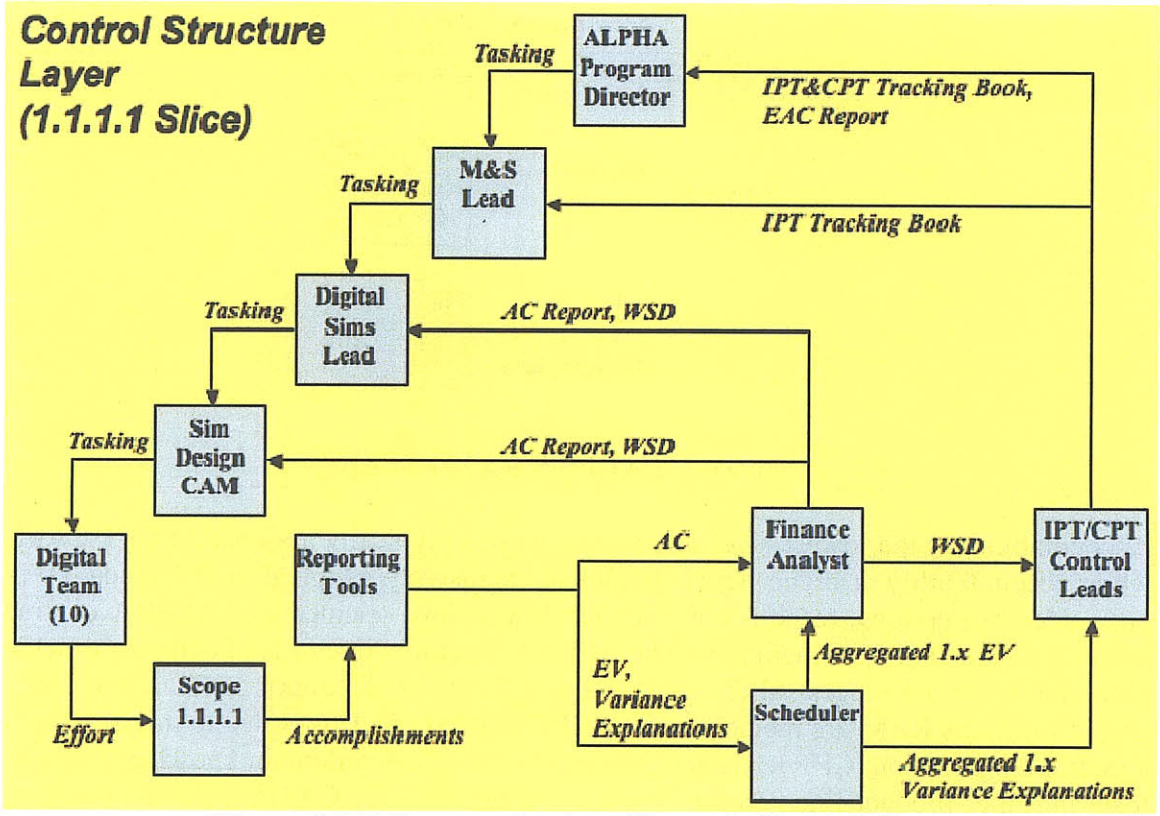

Figure 6.1-3 Communications Network (Partial View)

Figure 6.1-3 introduces a few new organizational roles: Scheduler, Finance Analyst, and IPT/CPT Control Leads. The Scheduler maintains the formal program schedule, tracks completion status for all contract scope elements, and monitors the critical path for potential risk. The Finance Analyst consolidates all reported EV and AC progress, aggregates the data for successively higher-level scope elements, and runs analysis reports to check for accuracy and threshold crossings. The IPT and CPT Control Leads review variance analysis reports for consistency, and prepare comprehensive monthly and quarterly progress reports as they apply to their respective organizations.

The reader may also have noticed a few unfamiliar data items in Figure 6.1-3. A brief description of each is as follows:

- Actual Cost (AC) Report – This weekly report captures the labor hours (i.e. AC) charged against each scope element and is tailored for each IPT. It is provided to all relevant sub-IPT leads and CAMs. It does <u>not</u> contain any information on EV.

- Work Status Document (WSD) – This monthly report captures all of the AC and EV progress, current period and cumulative, against all contract scope elements. It also contains the Finance Analyst's quantitative assessment of all SV and CV threshold crossings. This report is provided to all sub-IPT leads, CAMs, and IPT/CPT Control Leads. It does not contain any explanations for the scope element variances.
- IPT/CPT Tracking Book – This monthly report captures all of the AC and EV progress, current period and cumulative, against a single IPT or CPT. Variance explanations are provided along with the quantitative progress data. The IPT Tracking Book is provided to both the relevant IPT Lead and to the Program Director. The CPT Tracking Book is only provided to the Program Director.
- Estimate at Completion (EAC) Report – This monthly report captures the best available EAC data for all contract scope elements. This report is only provided to the Program Director.

In practice, this is how the control structure dynamically behaves:

- During the fiscal month, the Digital Team executes tasking provided by the Simulation Design CAM. AC is captured in a timecard system at the end of each week.
- Finance Analysts generate and distribute the weekly AC Report to the appropriate sub-IPT Leads and CAMs, for the previous week, within one to two business days.
- At the end of each fiscal month, the Simulation Design CAM captures EV progress in a reporting tool, which goes on to the Scheduler.
- The Scheduler aggregates all contract EV progress into a single report and provides it to the Finance Analyst.
- The Finance Analyst combines aggregated EV progress data with aggregated AC progress data, runs analysis scripts for variances and threshold crossings, compiles the results into the WSD, and provides the WSD to the CAMs, sub-IPT Leads and IPT/CPT Control Leads.
- CAMs write Variance Explanations for appropriate scope elements identified in the WSD. These explanations are then provided to the Scheduler.
- The Scheduler aggregates all contract Variance Explanations into a single report and provides the report to the IPT/CPT Control Leads.
- The IPT/CPT Control Leads take the WSD and the aggregated Variance Explanations and generate Tracking Books for the appropriate IPT Leads and the Program Director. Tracking Book meetings are typically held in the second week of each fiscal month.

6.2 Control Structure Layer Recommendations

In Section 4.3, we discussed that control systems are put in place to prevent or mitigate negative system behaviors in a process. And yet, poorly designed control systems can be ineffective or can themselves be the cause of controlled process issues. Design flaws can manifest themselves in the control system components, their behavior, their interfaces, and in their performance. Let us consider these control system-induced errors, along with the proposed specification and design, in the context of the example program.

The ALPHA program has a robust control structure layer. Mechanisms are in place for monitoring scope, communicating progress, forecasting completion, and reporting risks and losses. In fact, ALPHA meets all but one of the requirements defined in Section 5.2.2; this missing requirement is in the area of constraint enforcement and will be discussed in a moment. Despite meeting almost all of the proposed requirements, the author has identified three issues with ALPHA's control structure layer implementation. This section describes each of these issues and provides recommendations for improvement.

The requirement that the ALPHA control structure layer does not meet is R1.6, "*The control structure **shall** take corrective actions to prevent scope elements from transitioning to a state where the budget and schedule constraints have been violated.*" To be clear, ALPHA has processes in place to identify violations after they've occurred, and to predict areas where violations may occur in the future. But ALPHA has virtually no processes in place to prevent violations from occurring in the first place, nor any clear direction as to how to use risk prediction data to prevent violations. These failures can be attributed to three specific design issues found in the controllers and sensors of the ALPHA control structure layer. Let us now examine each of these issues.

6.2.1 Issue 1: Feedback Loop Design and Performance

The first issue with the ALPHA control structure involves the design and resulting performance of the organizational feedback loop. Recall that in a control system, a feedback loop measures data from a controlled process, converts that data into useful information, and provides the information to a controller for decision-making. In ALPHA, a suite of Reporting Tools fills the primary sensor component role. These Reporting Tools output information in the form of EV, AC and Variance explanations for a single scope element. This information then undergoes a series of secondary and tertiary conversions via the Finance Analysts, Schedulers, and IPT/CPT Control Leads. The end results of these conversions are the AC Reports, WSD, Tracking Books, and EAC Report.

The inputs and outputs of each sensor component within the ALPHA feedback loop are acceptable; the issue lies in the scope element measurement and reporting rates. Specifically, the fact that the rates at which EV and AC are tracked differs. Additionally, time delays incurred during the conversion processes leave significant gaps in current state information on scope elements and delay the issuing of corrective action from appropriate controllers. Put simply, the ALPHA sensor components are doing their jobs correctly, just not often enough nor quickly enough to prevent losses. For this discussion, we will assume work is only performed on business days, and that there are 20 business days in fiscal month. Table 6.2-1 below summarizes the effect that the current feedback loop design has on each piece of measurement information.

Table 6.2-1 Measurement Information Delays

Information	Scope (Days)	Delivery Day	Recipient	Delay Reason
AC Report 1	1-5	7	Sub-IPT Leads, CAMs	Timecards submitted at the end of each week, aggregation and conversion to AC takes approximately 1.5 days
AC Report 2	6-10	12	Sub-IPT Leads, CAMs	Timecards submitted at the end of each week, aggregation and conversion to AC takes approximately 1.5 days
AC Report 3	11-15	17	Sub-IPT Leads, CAMs	Timecards submitted at the end of each week, aggregation and conversion to AC takes approximately 1.5 days
AC Report 4	16-20	22	Sub-IPT Leads, CAMs	Timecards submitted at the end of each week, aggregation and conversion to AC takes approximately 1.5 days
WSD	1-20	23	Sub-IPT Leads, CAMs, IPT/CPT Control Leads	Milestone accomplishments submitted at the end of each month, aggregation, conversion to EV, and analysis takes 3 days
IPT/CPT Tracking Books	1-20	29	IPT Leads, Director	Variance Reports due on day 25, aggregation and analysis takes 4 days
EAC Report	1-20	35	IPT Leads, Director	ETC inputs due on day 30, aggregation and analysis takes 5 days

The faults in the feedback loop design should be clear at this point. AC is reported weekly, EV is reported monthly. Until both of these pieces of information are available, none of the derivative EVMS calculations can be performed. Additionally, the serial nature of Variance analysis and ETC reporting leads to a total delay of up to 15 days before predictive data on the full project scope reaches the program director.

In the interest of improving the design and performance of the ALPHA feedback loop, the author recommends that the program begin to collect EV data on a weekly basis, synchronized with the collection of AC data. The program should also begin to submit variance analyses and ETC inputs concurrently. The impact of not making these improvements is that controllers will be unable to prevent budget and scope violations from occurring, and will be unable to mitigate the effects when those violations occur.

6.2.2 Issue 2: State Definitions and Thresholds

The second issue with the ALPHA control structure involves the design of the organizational controllers. Recall from Section 4.1.1 that successful controller components contain accurate finite state machine representations of the controlled process. These models must account for all possible states of the controlled process, as well as all possible state transition events. ALPHA controllers do contain state models for each scope element, however it is the author's contention that these state models are poorly designed.

Every month, the ALPHA controllers receive a WSD containing budget and schedule information (e.g. CV, CV%, SV, and SV%) on all contract scope elements. ALPHA controllers are expected to analyze this information and make command decisions in order to provide appropriate tasking to their subordinates. This effort is equivalent to updating the controlled process state in an FSM. Table 6.2-2 below is a comparison between ALPHA model states and those of the thesis framework.

Table 6.2-2 State Model Comparison

State	ALPHA State Entrance Criteria	Thesis Framework State Entrance Criteria
Nominal	• *(CurrentSV >= -300)*, AND • *(CurrentSV% >= -10%)*, AND • *(CurrentCV >= -300)*, AND • *(CurrentCV% >= -10%)*, AND • *(CumulativeSV >= -1500)*, AND • *(CumulativeCV >= -1500)*	• *(CumulativeSV >= 0)*, AND • *(CumulativeCV >= 0)*
Warning	N/A	• *(0 > CumulativeSV >= RiskSV)*, OR • *(0 > CumulativeSV% >= RiskSV%)*, OR • *(0 > CumulativeCV >= RiskCV)*, OR • *(0 > CumulativeCV% >= RiskCV%)*
Risk	• *(CurrentSV < -300)*, OR • *(CurrentSV% < -10%)*, OR • *(CurrentCV < -300)*, OR • *(CurrentCV% < -10%)*, OR • *(CumulativeSV < -1500)*, OR • *(CumulativeCV < -1500)*	• *(RiskSV > CumulativeSV)*, OR • *(RiskSV% > CumulativeSV%)*, OR • *(RiskCV > CumulativeCV)*, OR • *(RiskCV% > CumulativeCV%)*

A few notes about Table 6.2-2 before we begin discussing the differences between the FSMs. First, ALPHA and the thesis framework have identical definitions for the *uninitiated, loss and completion states.* As such, these have been omitted from the table. Second, the risk threshold values shown in the ALPHA column are <u>not</u> the exact values used on the ALPHA program; they are for illustration purposes only. Third, all of the statements in both the ALPHA and thesis framework columns assume the following statements are true:

- *(Cumulative PV < BAC)*, AND
- *(Cumulative EV < BAC)*, AND
- *(Cumulative AC < BAC)*

There are quite a few differences between the ALPHA and thesis framework state definitions. Let us now examine these differences. To begin with, ALPHA utilizes current period variance information in its states, while the thesis framework does not. This was intentional on the part of the author as current period variances can be deceptive. Managers cannot issue accurate tasking based solely on what happened in the current period. Tasking can only be issued after considering a scope element's cumulative history and the associated impact of current period budget and schedule variances. Therefore, it is the author's recommendation that current period variance information should not be a factor in defining the state of the scope element.

Another difference found between the models is that ALPHA has common, program-wide thresholds for CV, CV%, SV, and SV%, while the framework has thresholds tailored to each scope element. This implies that every single scope element, independent of complexity, should be monitored in the same way. The author believes that this is a bad assumption. The reader may also have noticed that the only cumulative thresholds that ALPHA utilizes are *CumulativeSV* and *CumulativeCV*. But these numbers are essentially meaningless without also considering *CumulativeSV%* and *CumulativeCV%* thresholds. It is the author's recommendation that ALPHA tailor risk thresholds to each unique scope element, and begin incorporating *CumulativeSV%* and *CumulativeCV%* thresholds into their state models.

A final difference between the models is that ALPHA lacks a *warning* state for scope elements. Some may consider *nominal* and *risk* states to be sufficient to prevent violations. And to be fair, with the proper risk thresholds it is absolutely possible to manage budget and scope in this fashion. However the ALPHA program has set unusually high risk thresholds for both current and cumulative budget and schedule. Consider that a scope element could experience a cost overrun of as much as two person-months before tripping the current period risk threshold. Worse yet, a scope element could experience a cost overrun of as much as ten person-months before tripping the cumulative risk threshold. It is the author's recommendation that ALPHA introduce a warning state that triggers as soon as a scope element is in the red, and that ALPHA additionally tightens up all of its risk thresholds.

6.2.3 Issue 3: Controller Tasking

The third and final issue with the ALPHA control structure also involves the design of the organizational controller. In discussing the previous issue, we illustrated that ALPHA had a deficiency in the area of defining risks. But even if the program were to implement the author's suggestions, what should a manager do with risk information in order to prevent budget and schedule violations from being realized? The root of this issue is in the tasking, or commanded SPI and CPI, which managers issue to their subordinates.

Let us say for the sake of discussion that the CAM of scope element 1.1.1.1 has just been alerted by the Finance Analyst, that the Cumulative SV and SV% are perfect (i.e. both currently at 0), but the Cumulative CV and CV% are both indicating risk (e.g. -100 hours and -.25 respectively). How should a CAM task their team to avoid turning this budget

risk into a budget loss? To begin, the reality of the situation is that the team is spending too much money for too little accomplishment. It is intuitive to say that the team needs to start accomplishing more for the same or less money. In EVMS terms, the manager's resulting tasking needs to indicate an increased CPI. The real question is, how does a manager select the appropriate CPI value with which to task subordinates?

In its current design, the ALPHA program asks managers to issue tasking in the same fashion as the On-Off controller presented in Section 4.2.2.1. As a reminder, On-Off controllers only issue tasking signals of 0 or 1. In ALPHA's case, this would translate to telling team members to either do absolutely nothing or to work as hard as possible for as cheap as possible. This method of tasking only works for short duration and dire emergency situations. This method also does not take into account the team's past history of cost efficiency. ALPHA controllers need a more realistic methodology for providing the tasking their teams need to stay on track, which can also be tailored to the unique characteristics of each team.

In Section 5.3.1.4, the author provided one possible implementation to address this exact situation. The design called for a very simplistic P-controller that would be continuously updated as follows:

- Set *Commanded SPI* = $\dfrac{BAC - CumulativeEV}{BAC - CumulativePV}$
- Set *Commanded CPI* = $\dfrac{BAC - CumulativeEV}{BAC - CumulativeAC}$

This design was chosen as the primary implementation because it is intuitive and extremely easy to calculate. But based on our discussion of various control system designs in Section 4.2.2, the reader should be aware of a far more robust design that ALPHA could use for tasking – the PID-controller. The PID-controller issues signals that consider the past, present, and predicted future errors between a measured variable and a desired set point. Therefore it is the author's recommendation that ALPHA improve its tasking methodology by using the following PID-controller outputs:

- Set *CommandedSPI* to
  - $K_C(1 - CurrentSPI) + \tau_I \int_0^{\tau} (1 - CurrentSPI)dt + \tau_D \dfrac{d(1 - CurrentSPI)}{dt} + 1$
- Set *CommandedCPI* to
  - $K_C(1 - CurrentCPI) + \tau_I \int_0^{\tau} (1 - CurrentCPI)dt + \tau_D \dfrac{d(1 - CurrentCPI)}{dt} + 1$

The exact values for the three gain constants, as well as the integration window must be tuned to reflect the unique characteristics of each team and to balance or emphasize the proportional, integral, or derivative components as appropriate. Thus it is the author's recommendation that ALPHA implement PID-controller tasking for both SPI and CPI. The impact of not making this improvement is that controllers will continue to issue vague tasking that does not account for the realities of the team under their supervision.

7 Conclusion

This thesis opened with a quotation from Michael Sullivan, the Director of Acquisition and Sourcing Management of the Government Accountability Office. During his congressional testimony, Mr. Sullivan clearly presented an alarming trend of budget and schedule overruns on major Department of Defense (DoD) programs. He went on to propose a project-monitoring framework centered on three categories of data: *knowledge metrics*, *outcome metrics,* and *prerequisite indicators*. In short, these categories respectively represent scope progress gates, financial health reports, and pre-award viability analyses.

The framework proposed by Mr. Sullivan should be very successful at preventing ill-conceived programs from being awarded in the first place. His framework should also be successful at identifying program risks and losses after they have occurred. Unfortunately, as was stated in the thesis introduction, it is unlikely that any of the proposed monitoring tools will prevent or mitigate program losses on DoD or any other complex system programs. This inadequacy illustrates the need for the framework proposed in this thesis.

The specific objectives of this thesis were:

- To provide project managers with a mechanism to control risk within the scope of the work they oversee
- To provide individual contributors with a mechanism to control risk within the scope of the work they execute
- To clearly demonstrate how flaws in an organizational control structure can facilitate program losses
- To clearly demonstrate how a well designed organizational control structure can mitigate or even prevent program losses

The first two thesis objectives were addressed in Section 5. Within this section, a specification and design were proposed for a control structure that could be implemented on any complex system program, and in any lifecycle stage. The second two thesis objectives were met in Section 6. Within this section, a real DoD project was used to illustrate how control structure flaws allow both budget and schedule overruns to occur. Through the application of the thesis framework, the author was able to provide recommendations that would address each of the identified flaws.

The issues uncovered in the example program all centered on a lack of constraint enforcement on the part of the executing organizational control structure. "In systems theory and control theory, systems are viewed as hierarchical structures where each level imposes constraints on the activity of the level below it – that is, constraints or a lack of constraints at a higher level allow or control lower-level behavior [3, pg 7]." This lack of constraint enforcement is by no means unique to DoD projects. While there is certainly room for further research, the framework provided within this thesis can easily be applied to both new and existing complex systems projects, in order to reduce programmatic risk.

Appendix A: EVMS Guidelines

The following sections contain excerpts of the 32 formal EVMS guidelines from the ANSI EIA-748-B GEIA Standard EIA-748-B © 2008 Government Electronics and Information Technology Association.

Project Organization

- Define the authorized work elements for the program. A work breakdown structure (WBS), tailored for effective internal management control, is commonly used in this process.
- Identify the program organizational structure, including the major subcontractors responsible for accomplishing the authorized work, and define the organizational elements in which work will be planned and controlled.
- Provide for the integration of the company's planning, scheduling, budgeting, work authorization and cost accumulation processes with each other, and as appropriate, the program work breakdown structure and the program organizational structure.
- Identify the company organization or function responsible for controlling overhead (indirect costs).
- Provide for integration of the program work breakdown structure and the program organizational structure in a manner that permits cost and schedule performance measurement by elements of either or both structures as needed.

Planning, Scheduling, Budgeting

- Schedule the authorized work in a manner, which describes the sequence of work and identifies significant task interdependencies required to meet the requirements of the program.
- Identify physical products, milestones, technical performance goals, or other indicators that will be used to measure progress.
- Establish and maintain a time-phased budget baseline, at the control account level, against which program performance can be measured. Initial budgets established for performance measurement will be based on either internal management goals or the external customer negotiated target cost including estimates for authorized but undefinitized work. Budget for far-term efforts may be held in higher-level accounts until an appropriate time for allocation at the control account level. On government contracts, if an over-target baseline is used for performance measurement reporting purposes, prior notification must be provided to the customer.
- Establish budgets for authorized work with identification of significant cost elements (labor, material, etc.) as needed for internal management and for control of subcontractors.
- To the extent it is practicable to identify the authorized work in discrete work packages, establish budgets for this work in terms of dollars, hours, or other measurable units. Where the entire control account is not subdivided into work

packages, identify the far term effort in larger planning packages for budget and scheduling purposes.

- Provide that the sum of all work package budgets plus planning package budgets within a control account equals the control account budget.
- Identify and control level of effort activity by time-phased budgets established for this purpose. Only that effort which is not measurable or for which measurement is impracticable may be classified as level of effort.
- Establish overhead budgets for each significant organizational component of the company for expenses, which will become indirect costs. Reflect in the program budgets, at the appropriate level, the amounts in overhead pools that are planned to be allocated to the program as indirect costs.
- Identify management reserves and undistributed budget.
- Provide that the program target cost goal is reconciled with the sum of all internal program budgets and management reserves.

Accounting Considerations

- Record direct costs in a manner consistent with the budgets in a formal system controlled by the general books of account.
- When a work breakdown structure is used, summarize direct costs from control accounts into the work breakdown structure without allocation of a single control account to two or more work breakdown structure elements.
- Summarize direct costs from the control accounts into the contractor's organizational elements without allocation of a single control account to two or more organizational elements.
- Recover all indirect costs, which will be allocated to the program consistent with the overhead budgets.
- Identify unit costs, equivalent unit costs, or lot costs when needed.
- For EVMS, the material accounting system will provide for:
    - Accurate cost accumulation and assignment of costs to control accounts in a manner consistent with the budgets using recognized, acceptable, costing techniques.
    - Cost recorded for accomplishing work performed in the same period that earned value is measured and at the point in time most suitable for the category of material involved, but no earlier than the time of actual receipt of material.
    - Full accountability of all material purchased for the project including the residual inventory.

Analysis and Management Reports

- At least on a monthly basis, generate the following information at the control account and other levels as necessary for management control using actual cost data from, or reconcilable with , the accounting system:
    - Comparison of the amount of planned budget and the amount of budget earned for work accomplished. This comparison provides the schedule variance.

    - Comparison of the amount of the budget earned and the actual (applied where appropriate) direct costs for the same work. This comparison provides the cost variance.
- Identify, at least monthly, the significant differences between both planned and actual schedule performance and planned and actual cost performance, and provide the reasons for the variances in the detail needed by program management.
- Identify budgeted and applied (or actual) indirect costs at the level and frequency needed by management for effective control, along with the reasons for any significant variances.
- Summarize the data elements and associated variances through the program organization and/or work breakdown structure to support management needs and any customer reporting specified in the project.
- Implement managerial action taken as the result of earned value information.
- Develop revised estimates of cost at completion based on performance to date, commitment values for material, and estimates of future conditions. Compare this information with the performance measurement baseline to identify variances at completion important to company management and any applicable customer reporting requirements including statements of funding requirements.

Revisions and Data Maintenance

- Incorporate authorized changes in a timely manner, recording the effects of such changes in the budgets and schedules. In the directed effort prior to negotiation of a change, base such revisions on the amount estimated and budgeted to the program organizations.
- Reconcile current budgets to prior budgets in terms of changes to the authorized work and internal replanning in the detail needed by management for effective control.
- Control retroactive changes to records pertaining to work performed that would change previously-reported amounts for actual costs, earned value, or budgets. Adjustments should be made only for correction of errors, routine accounting adjustments, effects of customer or management directed changes, or to improve the baseline integrity and accuracy of performance measurement data.
- Prevent revisions to the program budget except for authorized changes.
- Document changes to the performance measurement baseline.

Bibliography

1. Allen, Thomas J. and Gunter Henn (2006). *The Organization and Architecture of Innovation*. Burlington, MA.

2. American National Standards Institute (ANSI) / Electronics Industry Alliance (EIA) Standard ANSI/EIA-748-B (2007).

3. Dulac, Nicolas, Brandon Owens and Nancy Leveson (2007). *Demonstration of a new dynamic approach to risk analysis for NASA's Constellation Program: MIT CSRL Final Report to the NASA ESMD Associated Administrator* [Electronic Version]. http://sunnyday.mit.edu/papers.html

4. Leveson, Nancy and Nicolas Dulac (2005). *Risk Analysis of NASA Independent Technical Authority* [Electronic Version]. http://sunnyday.mit.edu/papers.html

5. Leveson, Nancy and Kathryn Anne Weiss (2004). *Making Embedded Software Reuse Practical and Safe* [Electronic Version]. http://sunnyday.mit.edu/papers.html

6. National Defense Industrial Association (NDIA) Program Management Systems Committee (PMSC) Earned Value Management Systems Intent Guide (2009).

7. Sullivan, Michael J. (2009). *Defense Acquisitions: Measuring the Value of DoD's Weapons Programs Requires Starting with Realistic Baselines.* Government Accountability Office (GAO) Statement GAO-09-543T. Washington, D.C.