Incorporating System Complexity to Improve Development Cost Estimates

Dan Davis

CRM D0022312.A2/Final
April 2010

Approved for distribution: April 2010

[illegible]

Jino Choi
Cost Acquisition Team
Resource Analysis Division

Contents

This Page intentionally let blank.

Summary

The Principal Civilian Deputy to the Assistant Secretary of the Navy for Research, Development and Acquisition told us that there is anecdotal evidence to support the idea that the growing complexity of major Navy acquisitions drives cost growth in the major research and development programs. He wanted to know if CNA could demonstrate this more rigorously. Further, if complexity turned out to be a major contributor to the growth of development costs, he wanted us to see if there were some ways to incorporate complexity into generating better cost estimates as early as the preparation for the Milestone B review. This would give executives better information to make timelier decisions. Finally he also wanted us to see if there might be any implications for choosing to select a contractor as a lead system integrator.

Discussions with engineers and cost estimators within the Navy and with the chief naval engineer produced the following working definition of complexity. For a system under development, think of the components as nodes. But the nodes do not function independently. They must interact in specific ways either synchronously or in sequence with other nodes for the system to function properly. These interactions among the nodes define the system complexity.

Furthermore, the working hypothesis is that integrating and coordinating these interactions is costly. It is these costs that may not be adequately taken into consideration when the initial cost estimates are developed.

Armed with these insights, I focused on the development phase of a program as my level of analysis. I used the development contracts in the program as the nodes of my system definition. I used the maximum number of interfaces between pairs of nodes as a metric for system complexity.

However there is another well-known aspect of a complexity. In this view, a model of a phenomenon is complex if it consists of multiple

independent variables that interact with each other (interactions effects) and exhibit nonlinearities (second-order effects).

For example, suppose you have an aircraft program. For simplicity suppose that the major components are an airframe, the engines, and the avionics. This system has a level of system complexity already (three nodes). Now suppose the government changes its requirement for engine performance. The engines must now be able to deliver twice their original power (a change in the scope of the effort). This will also lead to possible changes in the airframe because of the changed aerodynamics. It may also lead to changes in the avionics. If these changes alter the weight of the air system, they may lead to a requirement for even more power. So this second level of complexity may drive costs even higher through interactions that ripple throughout the system. In fact the unintended consequences of this dynamic interaction may be exponential.

To test whether system complexity interacts with other cost drivers at this alternative level of complexity, I included scope growth and pure cost overrun in my working model. I used multiple regression analysis on a dataset of 176 completed programs to test the hypothesis.

The result was that complexity is a statistically significant driver of development cost growth. In addition, there are second-order effects. Further, complexity interacts with scope growth to increase cost growth. Interestingly, I found that for sufficiently low levels of scope growth, there may be some initial gains from specialization associated with complexity. However, these gains are soon exhausted and there is an upper bound on the level of complexity that could ever be optimal. This finding comports well with standard economic theory.

On the other hand, for sufficiently high levels of scope growth, it is always optimal to keep complexity to an absolute minimum. These findings generally support the intuition behind the working hypothesis, but in a more nuanced way.

Using this specific and empirically based functional relationship between program cost growth and complexity, I developed three applications. The first application is an Excel-based simulation that incorporates complexity in the model underlying its code. This simulation quickly updates cost estimates and risk estimates using the complexity of the program as a major input. This information is potentially available even before Milestone B. This application is available on the disc included with this paper.

A second application provides a way to determine the optimal span of control (number of contracts) for a program. This is the first time various management rules of thumb about spans of control have been empirically demonstrated.

A third application shows how to determine when choosing a single contractor as a prime or as a lead system integrator will be optimal. This has implications for selecting an acquisition strategy very early in the program, even before Milestone B.

Based on the encouraging results of my analysis, I recommend that the Navy use the simulation to get more realistic risk-based initial development cost estimates that include the effects of complexity. I also recommend that the Navy consider using the results to inform its selection of an acquisition strategy before Milestone B.

Finally, I recommend the Navy let us do further research on this topic. If a relatively robust set of technical readiness levels at the inception of a suitably large set of programs could be made available, I could model the relationship between initial technical maturity and the risk of future scope growth. This could dramatically improve my prototype model of complexity and the pilot simulation built on that model. This could make the pilot applications I developed even more useful to the Navy.

This page intentionally left blank.

What is complexity?

For every complex problem there is an answer that is clear, simple, and wrong. *H.L. Mencken*

Everything is simpler than you think and yet more complex than you imagine. *Anonymous*

Any fool can make things bigger, more complex ... It takes a genius and a lot of courage to move in the opposite direction. *Albert Einstein*

The estimation of the development cost of a weapons system by the Department of Defense (DoD) is very difficult. A compounding difficulty is that an initial estimate often turns out to be overly optimistic due to cost growth during the subsequent execution of the research and development (R&D) phase of the acquisition. Many believe that a major contributing factor to cost growth in the development phase of an acquisition is the complexity of the system being designed and built. If I can develop a reasonable working definition of complexity and I can gather data that reasonably measure complexity, I should be able to test this hypothesis.

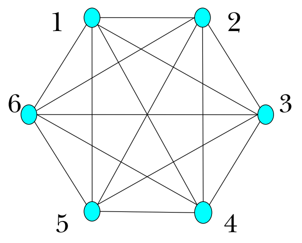
What is a reasonable definition of complexity? The Oxford English Dictionary defines something as complex if it consists of many different and connected parts. These connected parts interact to contribute to the performance of a desired function. More specifically, the parts must fit together in a well-matched way, and operate either in sequence or synchronously in very specific ways in order to generate the desired performance.

A simple example may serve to illustrate the concept of complexity. Consider a mousetrap. The basic parts of a mousetrap are: a flat wooden platform to serve as a portable base, a metal hammer, a spring to charge the trap, a sensitive catch, and a metal bar to connect to the catch. To work properly, these parts must be manufactured in specific ways to fit together in a specific design. The trap must then be baited and some parts must perform functions synchronously with other parts while other parts perform functions in a specific sequence in order to get the desired outcome: trapping a mouse. So, even something that seems relatively simple is more complex than first consideration might suggest.

This example demonstrates several characteristics of a complex system. First, when I speak of a complex device I am concerned with system behavior. That is to say, I am concerned with how multiple components work together in specific ways at specific times to produce a desired system output or performance. Second, the system output is a product of the synergy of the system: the whole is more than the sum of the parts. Third, each component is a product of specialization and works with other specialized components in the system. This specialization is a result of a comparative advantage that a component has while it performs a particular action in the system. Fourth, the components operate synchronously or in sequence with each other in specific ways. And fifth, the design, integration, and coordination of these components in a system are generally costly.

Engineers and scientists have begun to think about how to conceptualize the complexity of a system [1]. A conceptual design of how to think about a complex system is shown in figure 1 below.

Figure 1. A schematic of a complex system



In this example, the vertices represent the components of the complex system. Each component has a technical readiness level (TRL) that represents the technical maturity of the technology to manufacture that component. The maximum possible connections of these vertices represent the inter-relatedness of the components. For this example, consisting of six components, there are as many as 15 connections between pairs of components. These connections are sometimes call edges. In general, if there are *n* components (vertices), there are *n(n-1)/2* possible edges. Associated with these edges are hypothetical integration readiness levels (IRLs). Scientists are envisioning ways to assemble the TRLs and the IRLs in a matrix which can then be measured and normalized to generate an overall system readiness level (SRL) [1]. A stylized fact is that, when a system is more complex and has a lower extant level of technical maturity associated with its development, development costs rise higher than expected.

This architecture can then be applied at each level of design and manufacture. For example, we could think about the complexity of a component, itself made of subcomponents. At a higher level, we could think about the complexity of a system consisting of components. At a still higher level we could think about a 'system of systems' consisting of systems.

To make my investigation more tractable, I will use the R&D phase of a program as my level of analysis. This phase of a program may consist of one or more contracts, each of which defines work to be done to produce a component of the final product. The contracts represent the vertices on the schematic shown above in figure 1.

Using this basis for the definition and analysis of complexity, the plan of this paper is to describe my model, my basic methodology, and my assumptions. I plan to use this methodology and my assumptions to measure and test the hypothesis, which is: complexity is positively correlated with the growth of the cost of the R&D phase of a program above the initial cost estimate for that phase. I then describe my data that I used to test the hypothesis. I then show the results of my analysis. I then describe some applications suggested by my results. Finally, I offer recommendations and a conclusion.

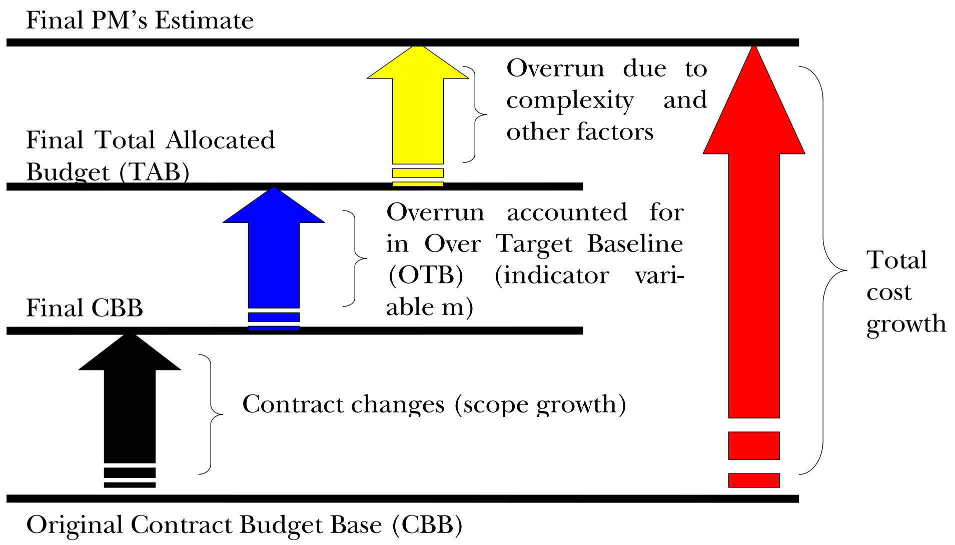
This page intentionally left blank.

Preliminaries

The model

I used as the basis of my model previous work that has been done [2]. A schematic of that model is shown in figure 2 below.

Figure 2. A schematic diagram of a cost growth model



Final PM's Estimate
Overrun due to complexity and other factors
Final Total Allocated Budget (TAB)
Overrun accounted for in Over Target Baseline (OTB) (indicator variable m)
Final CBB
Total cost growth
Contract changes (scope growth)
Original Contract Budget Base (CBB)

This schematic shows that development cost growth in a program can be broken down into contract changes, pure overrun, and some residual amount [2]. The total development cost growth is measured from the total development value of the program at its inception as measured by the total original development contract budget base (CBB). I assume that the sum of the final project manager's (PM) estimates for all the development contracts in a program is a good proxy for the final realized development cost of the program. Hence, the total development cost growth of the program is the difference between the final PM's estimates for all the development contracts in a program and the total of the original CBBs for the contracts.

I can measure the sum of the contract changes for all the contracts in a program by noting the difference between the sum of the original CBBs for all the contracts and the sum of the final CBBs for all the contracts. This difference in program value represents new work or scope in the contract. Hence, I assume that this difference is a good proxy for scope growth. That is to say, this difference is highly correlated with program scope growth in the development phase.

By definition, when the total allocated budget for a contract (TAB) is different from the CBB, there has been an over-target baseline (OTB). This difference between TAB and CBB represents pure cost overrun. I measure the difference between the total final TABs for all the development contracts in the program and the total final CBBs. This difference represents pure cost overrun in the development phase of the total program. I represent this in my model with an indicator variable *m* which takes the value of 1 if there has been at least one OTB in the program, and a value of 0 if there have been no OTBs in any development contracts in the program.

In my model, I assign the cause of the residual difference between the final PM's estimate and the final TAB to complexity and other factors. This will become the basis for testing the hypothesis that program complexity is positively correlated with program cost growth.

At this point, I want to introduce another aspect of complexity. There is a large body of literature on complexity and chaos theory that conceptualizes complexity as the presence of nonlinearities in the system, in addition to the previously described connections, feedbacks, and interactions between components in the system [3, 4, and 5]. To capture this nonlinear aspect of complexity, I include interaction terms and second-order effects in my model [6] to see if they are significant.

I also want to point out that I also measured and tested other potential models of development cost growth in a program. The potential predictors in these models included: program duration, program schedule growth, service component (Army, Air Force, or Navy), and all of their associated possible interactions. None of these variables proved to be statistically significant explanations of cost growth in the development phases of programs; hence, these models were discarded.

As a result, my model describes R&D program cost growth as a function of:

- scope growth
- pure cost overrun indentified with an OTB
- complexity
- interaction terms
- second-order effects and
- unobserved random factors

A mathematical description of my model is in Appendix A.

Methodology

With a model in hand, I planned to use multiple regression analysis to analyze the data [6]. Before proceeding, I further refined my measurement of the variables by converting differences to percentage changes, so that program growth metrics were measured in rates of growth over the life of the program. For example an observed program scope growth of 0.2 represents a 20 percent increase in the scope contained in the program from its original work content.

The various program growth rates were calculated as follows:

- total program cost growth is: (total PM's final estimate – total original CBB)/total original CBB
- total contract change or scope growth is: (total final CBB – total original CBB)/total original CBB
- pure cost overrun is measured by an indicator variable which

- takes the value of 1 if there has been at least 1 OTB
- takes the value of 0 if there have been no OTBs

I measured complexity by associating the contracts in the development phase of the program with component nodes in the system. I then calculated the number of potential connections or edges in the program. To prevent problems associated with division by zero when calculating elasticities later in the analysis, I ultimately measured complexity as the number of edges plus 1. The theory I am testing is that complexity causes additional integration and coordination activities that are costly and contributes to overall program development cost growth.

I also wanted to include nonlinearities in the model. In addition to the inherent nonlinearity associated with my measure of complexity, I also included the interaction of scope growth and complexity as an additional nonlinearity. The theory is that scope growth changes the value of contract nodes and complicates the integration and coordination of the contracts as program development continues. This added complication contributes to overall program cost growth.

A further nonlinearity I included in the model is a second-order effect. I included the 'complexity-squared' variable as a possible additional variable affecting cost growth. The theory is that there are possible gains from specialization in the components of a complex system. Including this squared term allows me to test for this possibility.

Finally, I included unobserved factors contributing to program cost growth. I modeled this unobserved part of cost growth as a random, i.i.d. (identically and independently distributed) variable. I made the usual assumption that this random variable is normally distributed with a mean of zero and a homoscedastic (constant and finite) variance. The model is shown in mathematical terms in appendix A.

The data

I used data from the Contract Analysis System (CAS) from DoD, available from the Director, Acquisition Resources and Analysis. This database included 220 programs with complete data. These programs were comprised of 1400 contracts. There were a total of 16,854 records to sort through. The data spans 1970 to 2006.

I found 176 programs with complete and accurate data that contained at least one development contract. This dataset was what I used to analyze the model of complexity and program cost growth in the development phase. I included only the development contracts for each program in my analysis. The data that I used in my analysis is in Appendix C.

The programs were evenly distributed among the services. Of these 176 programs, there were

- 57 Navy programs,
- 61 Army programs, and
- 58 Air Force programs.

I did pair-wise comparisons of the services' data for development cost growth. I did two non-parametric tests on each pair to see if the differences in the samples were significant. The results are shown in table 1.

Table 1. Test results of pair-wise sample comparisons (ten-percent level of significance)

Null hypothesis	Test	p-value	Result
Navy sample program development phase cost growth comes from same distribution as Army cost growth	Kolmogorov-Smirnov test	0.057	Reject null
	Wilcoxon rank-sum test	0.079	Reject null
Navy sample program development phase cost growth comes from same distribution as Air Force cost growth	Kolmogorov-Smirnov test	0.010	Reject null
	Wilcoxon rank-sum test	0.007	Reject null
Army sample program development phase cost growth comes from same distribution as Air Force cost growth	Kolmogorov-Smirnov test	0.661	Fail to reject null
	Wilcoxon rank-sum test	0.476	Fail to reject null

The results of the pair-wise comparisons and the expected rank-sums suggest that the Navy has generally lower development program cost growth than the other two services, at the ten-percent level of significance. However, later testing of my full regression model found that service component was not a significant predictor of program development cost growth, at the five-percent level of significance.

The average program cost growth was 0.68 (68 percent). The median cost growth was 0.41 (41 percent). Of the 176 programs

- 166 programs had a positive total cost growth,
- 6 programs had a negative total cost growth, and
- 4 programs had zero total cost growth.

The average program scope growth was 0.47 (47 percent). The median scope growth was 0.21 (21 percent). Of the 176 programs

- 161 programs had a positive scope growth,
- 8 programs had a negative scope growth, and
- 7 programs had zero scope growth.

The average program schedule growth was 0.22 (22 percent). The median schedule growth was 0.04 (4 percent). Of the 176 programs

- 102 programs had a positive schedule growth,
- 10 programs had a negative schedule growth, and
- 64 programs had zero schedule growth.

In the dataset of 176 programs, there were 135 programs with no OTB. There were 41 programs with at least one OTB.

In the dataset of 176 programs the minimum number of development contracts observed in a program was 1. The maximum number of contracts observed in the development phase of a program was 12. The average number of contracts was 3. The median number of contracts was 2. There were a total of 515 development contracts in the dataset.

This page intentionally left blank.

Results

I estimated the model of R&D program cost growth as a function of scope growth, cost overrun, complexity, the interaction of complexity and scope growth, and complexity squared. Specifically I estimated the following equation: *cost growth = b_0 + b_1 * scope growth + b_2 * cost overrun indicator + b_3 * complexity + b_4 * complexity*^{2} *+b_5 * scope growth * complexity*. For the mathematics of the model, see appendix A.

For the model, my results are shown in table 2 below.

Table 2. Model results

Model	n = 176	F(5,170) = 341.70	p = 0.0000	R^2 = 0.9095	Adjusted R^2 = 0.9068

The F-statistic and the very low p-value for the model are evidence that the relationship between program development cost growth and the independent variables of the model is statistically significant, at the five-percent level of significance. The coefficient of determination (R^2) suggests that the model explains 91 percent of the variation in the data.

I did sample correlations among the independent variables of the model. The results are shown in table 3.

Table 3. Sample correlation matrix

	Scope growth	**Indicator variable for OTB**	**Complexity**	**Scope growth times complexity**
Scope growth	1.000			
Indicator variable for OTB	0.111	1.000		
Complexity	-0.051	0.241	1.000	
Scope growth times complexity	0.519	0.141	0.391	1.000

Statisticians have developed a rule of thumb that if the absolute value of the sample correlation coefficient is greater than 0.7 for any two independent variables, multicollinearity is a potential problem [6]. The evidence in the correlation matrix suggests that multicollinearity is not a problem in my model.

The specific effects of each of the cost growth drivers in the model are summarized in table 4 below.

Table 4. Regression analysis results

Variable	Coefficient	p value
scope growth (**scg**)	0.969 (0.0343)	0.000
cost overrun indicator variable (**m**)	0.279 (0.0554)	0.000
complexity (**e**)	-0.02238 (0.00717)	0.002
complexity^2 ($\mathbf{e}^2$)	0.00024456 (0.0001145)	0.033
interaction of scope growth and complexity (**scg** * **e**)	0.0306 (0.00606)	0.000
constant (**k**)	0.1895 (0.03776)	0.000

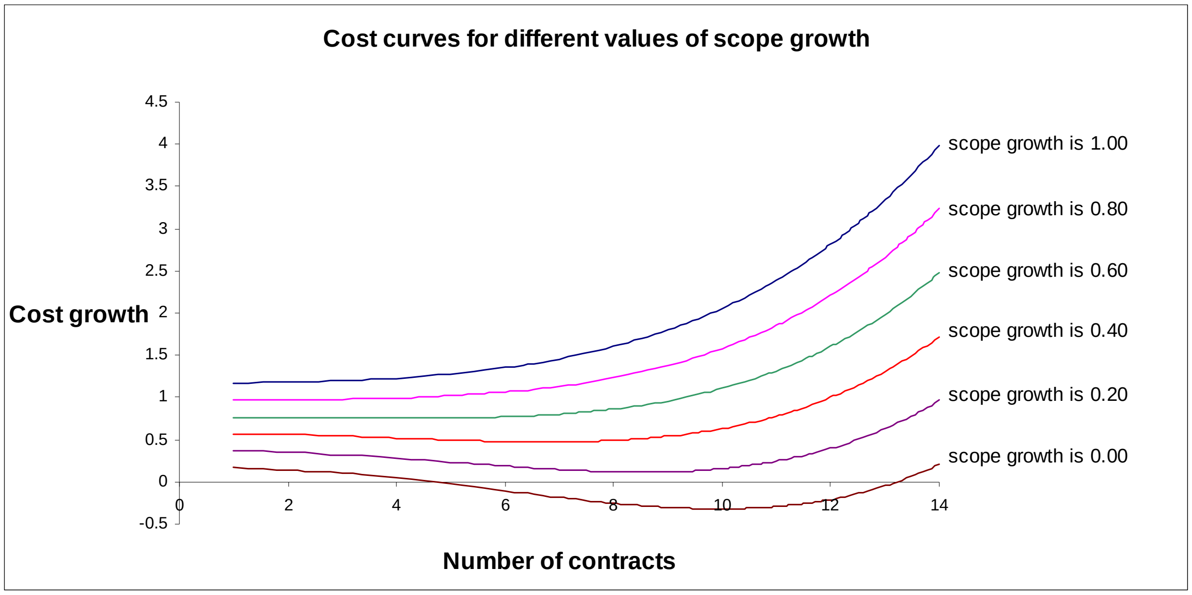
The p-values are very low for each estimated coefficient. This suggests that all the coefficients are statistically significant at the five-percent level of significance. The positive signs on the coefficients for scope growth and cost overrun comport with expectations. Our theory and our intuition are confirmed. Scope growth and cost overrun are positively and significantly correlated with overall program development cost growth.

The positive coefficients on the complexity-squared term and the interaction term suggest that our theory is correct. These positive coefficients suggest that complexity by itself and its interactions with scope growth do indeed contribute to development cost growth.

The interpretation of the negative sign on the simple complexity variable (*e*) suggests that there are gains to be had from complexity as a result of specialization. These gains actually reduce cost growth, on average. This is a slightly surprising finding and requires more exploration.

To see the relationship between cost growth and both scope growth and complexity, I graphed the functional relationship in figure 3 below.

Figure 3. The relationship of cost growth, scope growth, and complexity



Since the measure of complexity is directly and monotonically related to the number of contract nodes in a program system, I measured the number of contracts on the horizontal axis. All the curves represent cost growth curves holding the cost overrun at $m = 0$. This is without loss of generality since a similar graph with cost overrun held constant at $m = 1$ would just shift all the curves up by a small amount equal to the coefficient for the cost overrun variable shown in table 4. The basic intuition of the graph would be exactly the same.

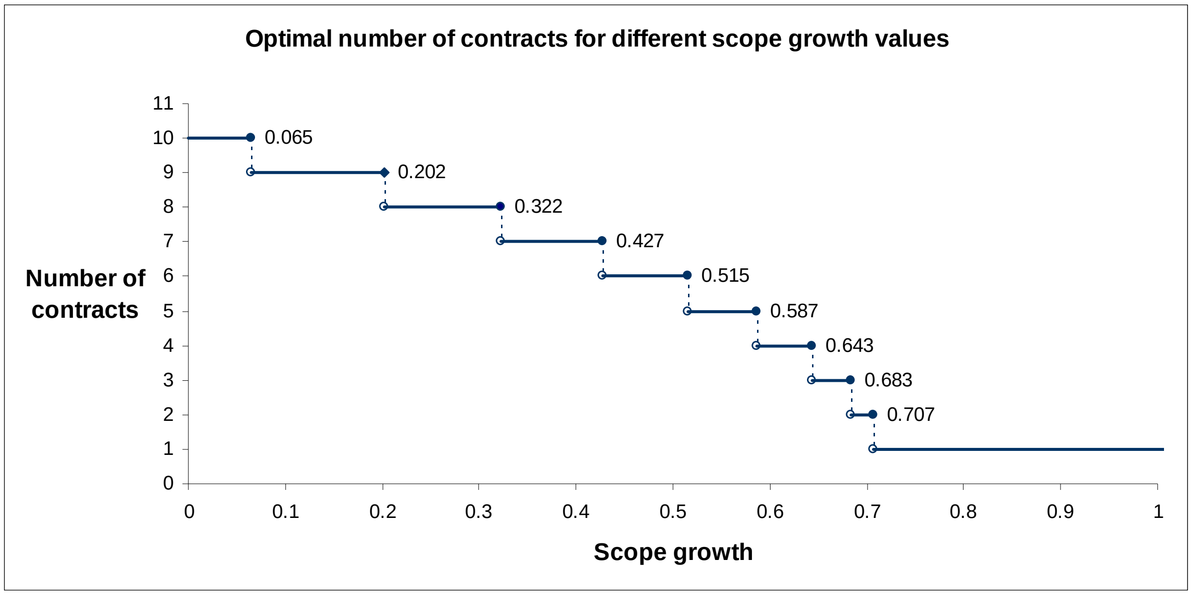
The first thing to note is how cost growth is related to the number of contracts (program complexity). Looking at the curve when scope growth is held constant at 0.2, notice that as you move from left to right the cost falls then rises with the increasing number of contracts (complexity) in the program. This suggests that at a low level of scope growth, cost growth initially falls as complexity increases. This contradicts my working hypothesis. This only lasts up to a point however. Then as the number of contracts continues to rise, the cost growth rises. This eventual increase of cost growth with complexity supports my working hypothesis.

A possible explanation of this apparent contradiction is that there may be initial gains from specialization resulting from comparative advantages associated with more specialized components in a system. However, these gains are eventually exhausted. At this point the additional costs of integration and coordination inherent in a complex system begin to predominate. From this point on, an increase in complexity is correlated with an increase in cost growth.

Second, note the changes in the shapes of the curves as scope growth increases. The minimum point for cost growth on a particular curve shifts to the left as you successively move upward from one curve to the next. Eventually, on the cost curve associated with a scope growth of 0.8 for example, there is no reduction in cost growth. Cost growth rises monotonically for any increase in complexity. This supports my working hypothesis.

From an inspection of the cost curves shown in figure 3, it becomes apparent that the results of the analysis of my model allow me to determine an optimal level of complexity for any given level of realized scope growth. These optimization results are shown in figure 4 below.

Figure 4. Optimization results



To interpret this graph, pick a level of scope growth. Let us say scope growth equals 0.33 (33 percent). To determine the optimal number of contracts for such a program, read vertically from 0.33 until you intersect the step function (since contracts or components can only exist in whole numbers). The graph shows that the optimal number of contracts in this case is 7. Note that for very high levels of scope growth (greater than 0.707), the optimal number of contracts from the government's perspective is one. For very low levels of scope growth (less than or equal to 0.065), the optimal number of contracts is 10. In no case is the optimal number of contracts ever above 10. This suggests that there is an upper limit on the ideal number of contracts, and the ideal number itself depends on the level of scope growth realized.

Now the problem with this is: you don't know the level of scope growth that will occur in a program in advance. However, you should be able to form a realistic expectation. Perhaps a good basis for forming this expectation is the technical maturity you have as you begin your program. It seems to make sense that a higher level of technical maturity (TRL) would cause you to expect a lower scope growth. This could inform your acquisition strategy as you choose the number of contracts to use to manage the program. On the other hand, if the technical maturity (TRL) is low as you begin your program, you might reasonably expect a large level of scope growth in development. This, too, might guide your acquisition strategy. I will explore this more fully in the applications section of this paper.

In general though, my results comport well with standard economic theory. Depending on the level of scope growth realized, cost growth may decline with an increase in complexity as gains from specialization predominate. The gains are soon exhausted, however. After that point, cost growth increases monotonically with complexity.

Having thoroughly analyzed the results with regard to statistical significance and the sign of the coefficients, I now turn to analyzing the relative size of the coefficients. To do this I will use the concept of elasticities. The mathematics of elasticities is shown in appendix B.

To quickly review the intuition behind the idea of elasticities, suppose the elasticity of cost growth with respect to scope growth were 0.6 at a particular point. That would mean that from that point, a one percent increase in scope growth would on average lead to a 0.6 percent increase in cost growth. The elasticity is a unit-less number that allows us to compare elasticities with respect to different independent variables.

I calculated arc elasticities at different points for the elasticity of cost growth with respect to complexity, for a given level of scope growth. Graphs of families of elasticity curves are shown in figures 5 and 6 below.

Figure 5. Elasticities of cost growth with respect to complexity (m = 0)

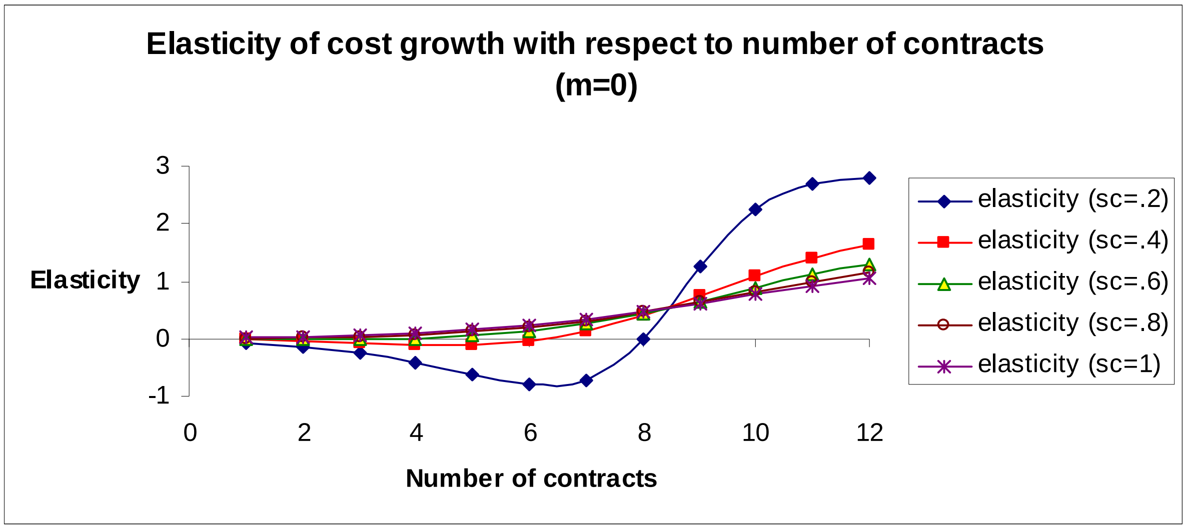
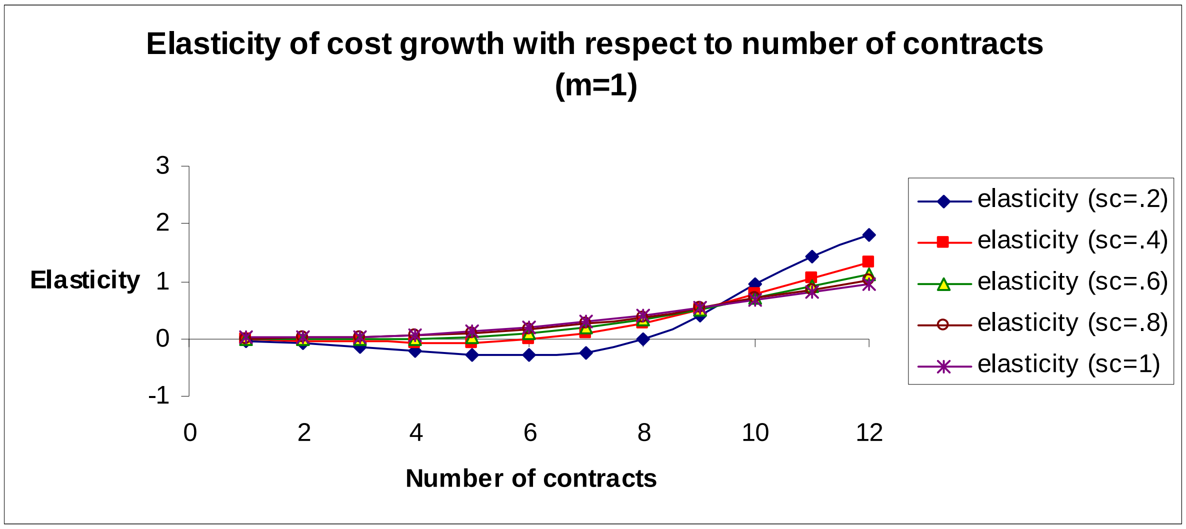


Figure 6. Elasticities of cost growth with respect to complexity (m=1)



The above graphs show that for low levels of scope growth (for example scope growth equal to 0.2), the elasticity of cost growth with respect to complexity may be low and even negative at low levels of complexity. However, when I look at small increases in complexity from larger initial levels of complexity (points further right on the graph), the elasticity of cost growth with respect to complexity grows.

For large levels of scope growth (for example scope growth equal to 0.8), the elasticity of cost growth with respect to complexity may be low but it is always positive, for low levels of complexity. Again, as I move further to the right on the graph, the elasticity of cost growth with respect to complexity grows. This analysis is true whether $m = 0$ or $m = 1$.

I also calculated arc elasticities at different points for the elasticity of cost growth with respect to scope growth, for a given level of complexity (number of contracts). Graphs of families of elasticity curves are shown in figures 7 and 8 below. The variable *c* in the graphs represents the number of contracts in the program (a measure of complexity).

Figure 7. Elasticities of cost growth with respect to scope growth (m=0)

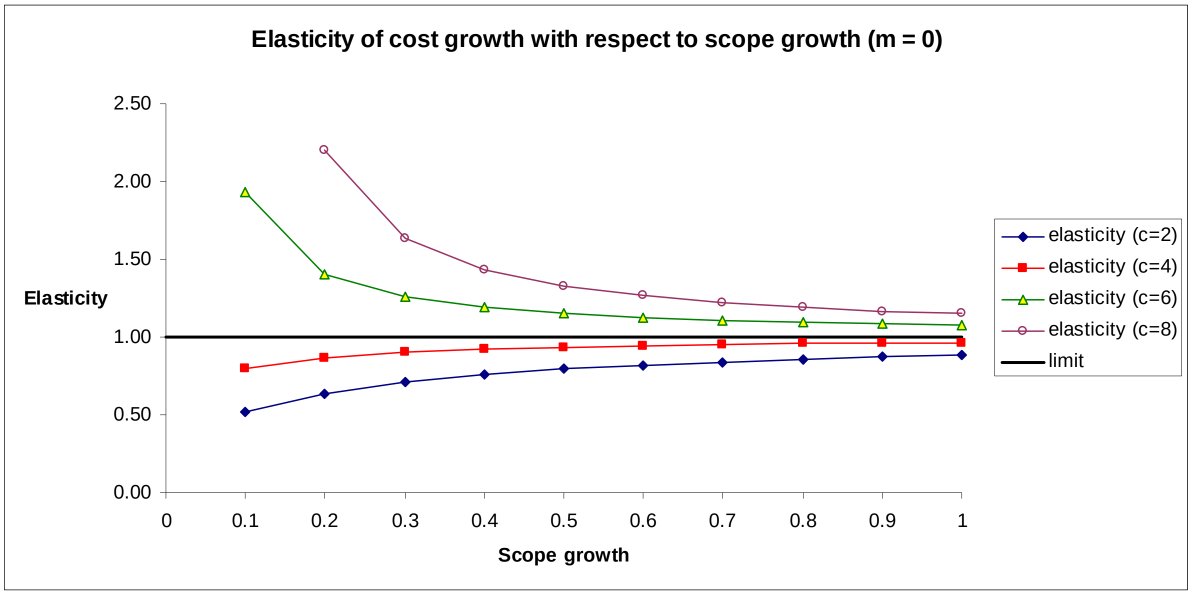
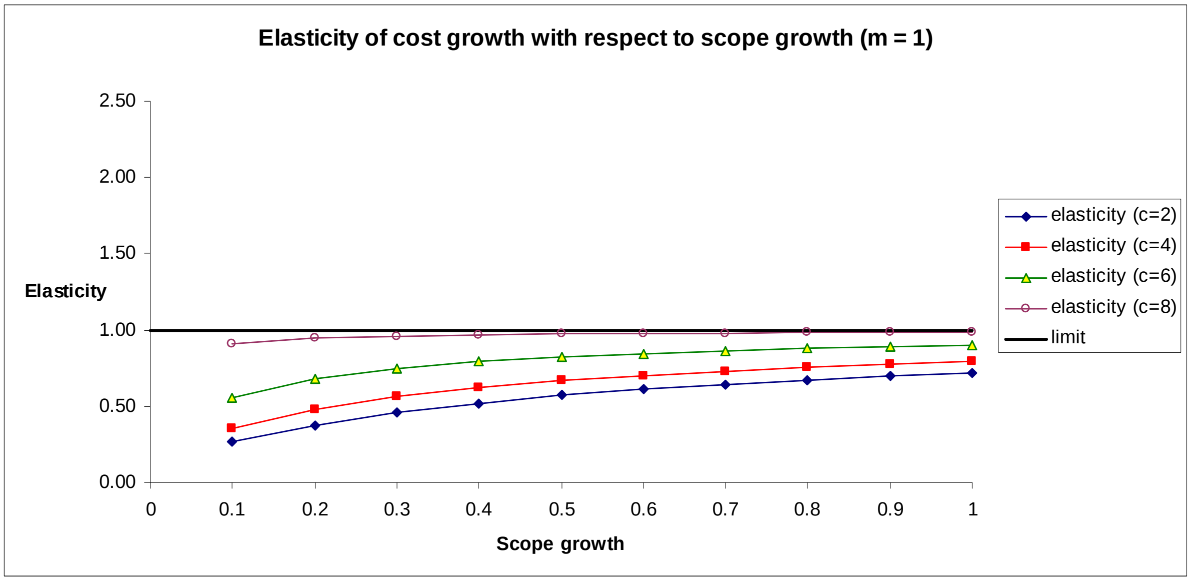


Figure 8. Elasticities of cost growth with respect to scope growth (m=1)



The above graphs show that for low levels of scope growth, the elasticity of cost growth with respect to scope growth is always positive. As you move further to the right on the graph, the elasticity of cost growth with respect to scope growth approaches 1 in the limit. This analysis is true whether $m = 0$ or $m = 1$.

I computed the semi-elasticity of cost growth with respect to cost overrun, since I measure cost overrun with an indicator variable. I evaluate that elasticity as a point elasticity evaluated at the median cost growth observed in my dataset. The semi-elasticity of cost growth with respect to cost overrun is 0.68. This is positive but relatively small. If a program reports cost overrun (as opposed to having no reported cost overrun), the total cost growth will on average be 0.68 percent higher due to the cost overrun, all other factors being equal.

I should add a caution when interpreting the elasticities. Since cost growth and scope growth are measured as percents, the elasticities I have noted above reflect a percent of a percent.

Nonetheless, the results above show that in most cases the elasticities have the expected signs. Furthermore, the results show that in some cases the elasticity of cost growth with respect to complexity may be large (greater than 1).

In general my model and the results of applying that model to data are empirical evidence that increases in complexity either eventually or immediately lead to cost growth in programs. The effects of complexity impose an upper bound on the optimal number of contracts for a program. Further, the effects of increases in complexity on cost growth can be quite large. Finally, my model shows an additional element of complexity in the interaction effect between complexity and scope growth and the second-order effect of complexity-squared.

With the qualifications noted in my analysis above, the evidence suggests that my working hypothesis that increased complexity is correlated with increased R&D program cost growth is indeed true. What is more, I now have a model that I can with some confidence use to improve initial cost estimates, taking into account complexity and the risk of scope growth. These applications are the subject of my next section.

This page intentionally left blank.

Applications

Simulation

Armed with a model of cost growth that appears to describe the data well, I can construct a simulation to improve on the realism of an initial cost estimate of an R&D program. I take as given that the cost estimation community is quite good and the rigor of their estimates assure, that *ceteris paribus*, their initial program cost estimates are good. Given that estimate and since I want to develop a management application for use at or before Milestone B that simulates uncertain future development scope growth and pure cost overrun, I propose to use Monte Carlo simulation to provide a better feel for the risk in that estimate. With this picture of future risk, the decision-maker should have a better understanding on which to base his expectations and to choose a level of acceptable risk.

The simulation uses the model estimated in the last section. The simulation works in Excel and asks the user for two inputs: the initial independent cost estimate and the number of contracts planned in the acquisition strategy for managing an R&D program.

I model unknown future scope growth as a random variable. My sample data suggests that it is likely distributed lognormally. I do two non-parametric tests and a q-plot to test this. The results are shown in table 5 and figure 9 below.

Table 5. Test results of lognormal hypothesis for distribution of scope growth

Null hypothesis	Test	p-value	Result
Sample program development phase scope growth is distributed lognormally	Kolmogorov-Smirnov test for lognormal distribution	0.089	Fail to reject null
	Shapiro-Wilk test for 3-parameter lognormal distribution	0.113	Fail to reject null

Figure 9. Q-plot for sample scope growth

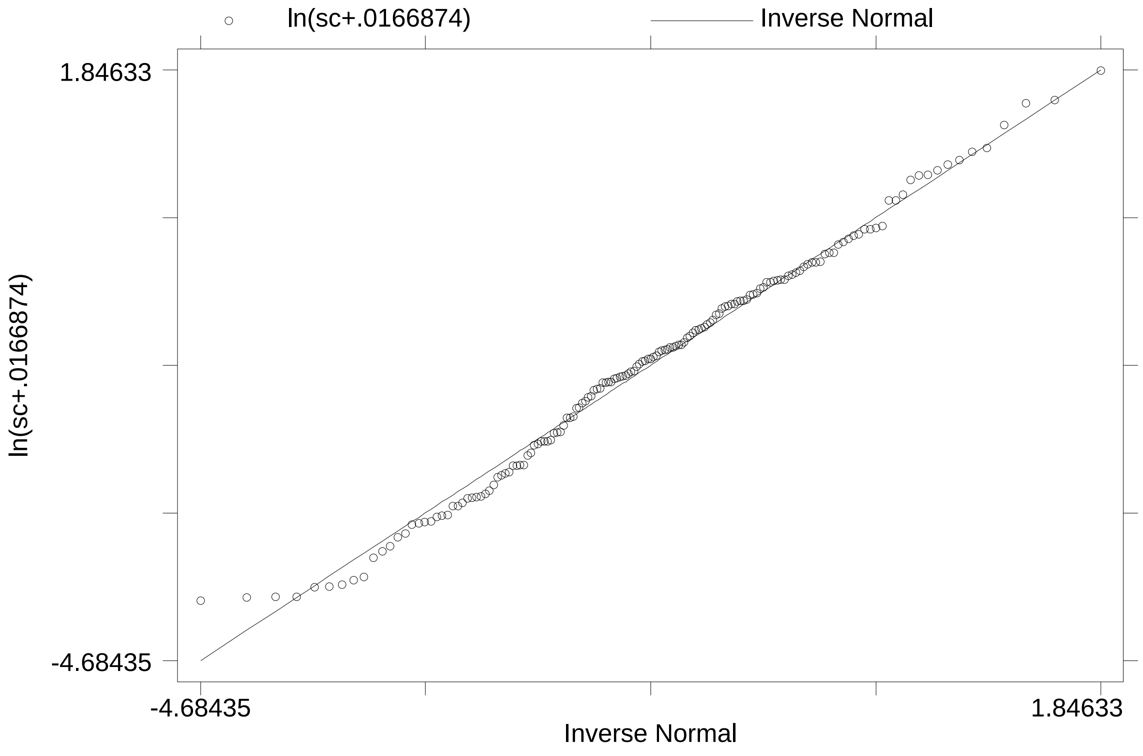


Table 5 shows that two non-parametric tests suggest that I cannot reject the hypothesis that the sample scope growth is generated by an underlying lognormal distribution, at the five-percent level of significance. The q-plot shows that the sample data is generally close to the straight line of the inverse curve. This suggests visually that scope growth is distributed lognormally.

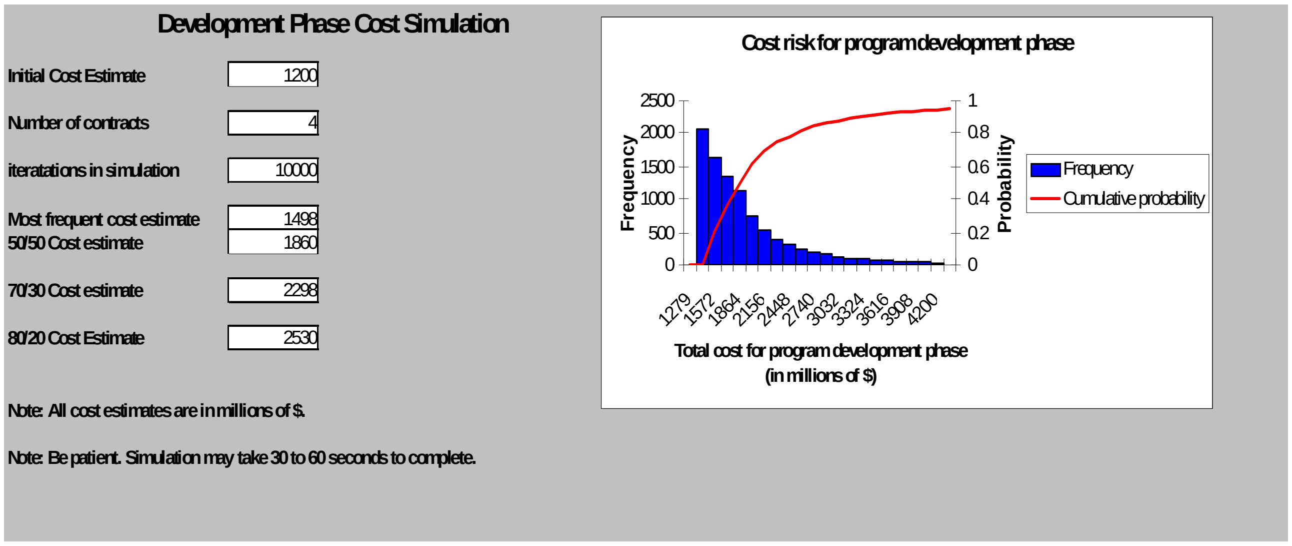
So I draw from a lognormal distribution in my simulation to get scope growth. I draw from a uniform distribution on the unit interval to generate an indicator value (0 or 1) for cost overrun.

I now have all I need to run one iteration of my simulation to generate predicted R&D program cost growth. I input the data into my model, and repeat as many times as I like (the default chosen in the simulation I constructed is 10,000 iterations).

I ran several test simulations to calibrate the model and the parameters of the random distributions from which I got my random draws. The purpose of this calibration was to generate simulation data that replicated closely the sample statistics of my dataset of 176 programs.

Having completed the calibration, I was done constructing the simulation. An example output is shown in figure 10 below.

Figure 10. Development phase cost simulation output



Development Phase Cost Simulation

Initial Cost Estimate	1200
Number of contracts	4
iteratations in simulation	10000
Most frequent cost estimate	1498
50/50 Cost estimate	1860
70/30 Cost estimate	2298
80/20 Cost Estimate	2530

Note: All cost estimates are in millions of $.

Note: Be patient. Simulation may take 30 to 60 seconds to complete.

To begin the simulation, the user should press the "perform simulation" button. This causes the simulation to present to the user a series of message boxes asking for user input. For this example, I input that I wanted to run 10,000 iterations of the simulation (the default entry). When prompted, I then input an initial independent cost estimate for the R&D program of $1.2B. When prompted again, I then input that the acquisition strategy envisioned using four contracts to manage the program. The simulation then took a little over 30 seconds to run and produced the display shown above.

The frequency chart gives a visual picture of the likelihood of different ranges of eventual program costs that includes the effect of complexity, the possibility of scope growth, and the possibility of a cost overrun. The most frequently observed total realized R&D program cost was approximately $1.498B in this case.

The simulation also displays a cost risk curve in red which is a cumulative distribution curve associated with the sample data generated by the simulation. This allows you to calculate risk-adjusted R&D program costs that take complexity into account.

The 50/50 cost in this case was approximately $1.86B. This means there is a 50 percent chance the eventual cost will be below $1.86B and a 50 percent chance the cost will be above $1.86B.

The 70/30 cost is $2.298B. This means there is a 70 percent chance the eventual cost will be below $2.298B and a 30 percent chance the eventual cost will be above $2.298B.

The 80/20 cost is $2.53B. This means there is an 80 percent change the eventual cost will be below $2.53B and a 20 percent change the cost will be above $2.53B.

This gives the decision-maker a robust and realistic picture of the cost risk in the R&D program. The simulation incorporates the aggregate behavior of past R&D programs and the role of complexity in contributing to cost growth. This application is built on my analysis and the results of that analysis described in the last section. Of course the simulation can be modified to provide different displays and information to suit the needs of the decision-maker. This particular simulation is only a prototype to demonstrate the potential of simulations to inform the decision-maker and to show how the inclusion of program complexity in the model may improve the early cost estimates.

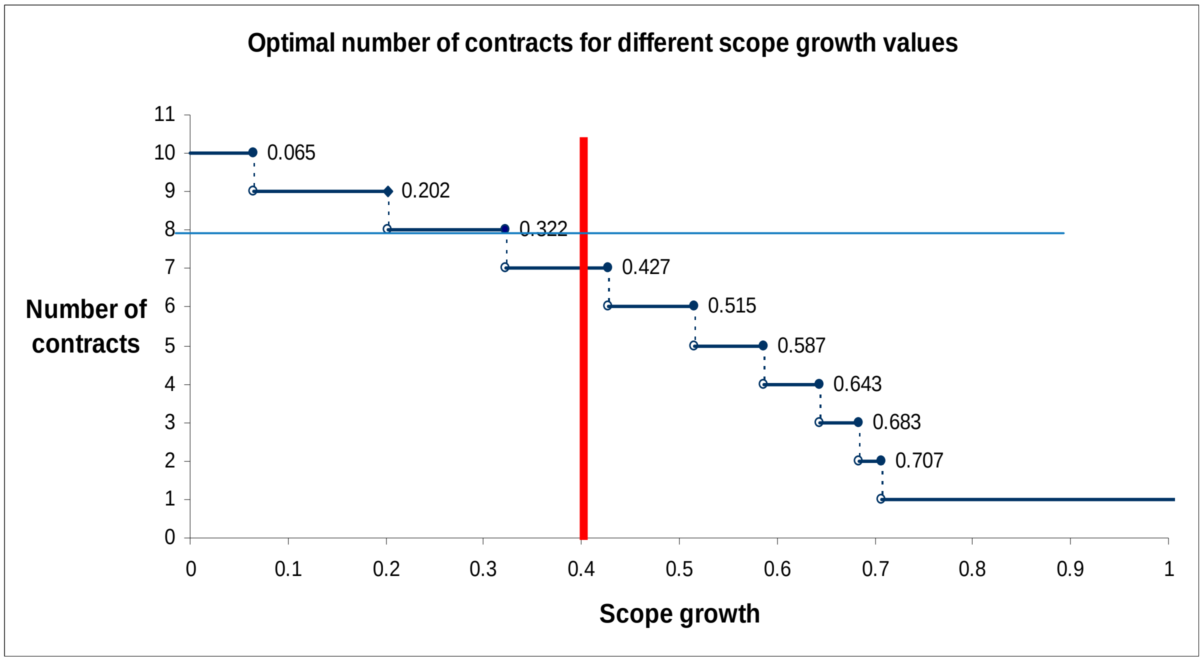
Span of control

Another application of my analysis of complexity and cost growth relates to span of control. I have always heard that the effective manager or leader of an organization should limit his span of control to 3 to 7 subordinates reporting directly to him. The Economics Nobel laureate Herbert Simon has noted that administrative efficiency in an organization depends (among other things) on the trade-off between specialization and span of control. He further noted that proponents suggest optimal spans of control of 3, 5, 7 or even as many as 11. Simon also observed that there was little empirical support for any of these rules of thumb [7].

In a similar vein, the Economics Nobel laureates Ronald Coase [8] and Oliver Williamson [9] discuss the issue of the boundary of the firm. They suggest there are transaction costs associated with going outside the firm to obtain inputs for production. On the other hand there are bureaucratic inefficiencies and potential losses of the gains from specialization if a firm vertically integrates to make its inputs for production internally. These trade-offs are similar to the problem the government faces when deciding on an acquisition strategy for the architecture of contracts used to manage a program.

My analysis of complexity and cost growth and the results of that analysis suggest, *pace* Simon, that there is empirical support for the span-of-control issue. To illustrate how the results may be used to answer the question of the ideal span of control, I will use the diagrams in figures 11 and 12 below.

Figure 11. Complexity and span of control: example 1

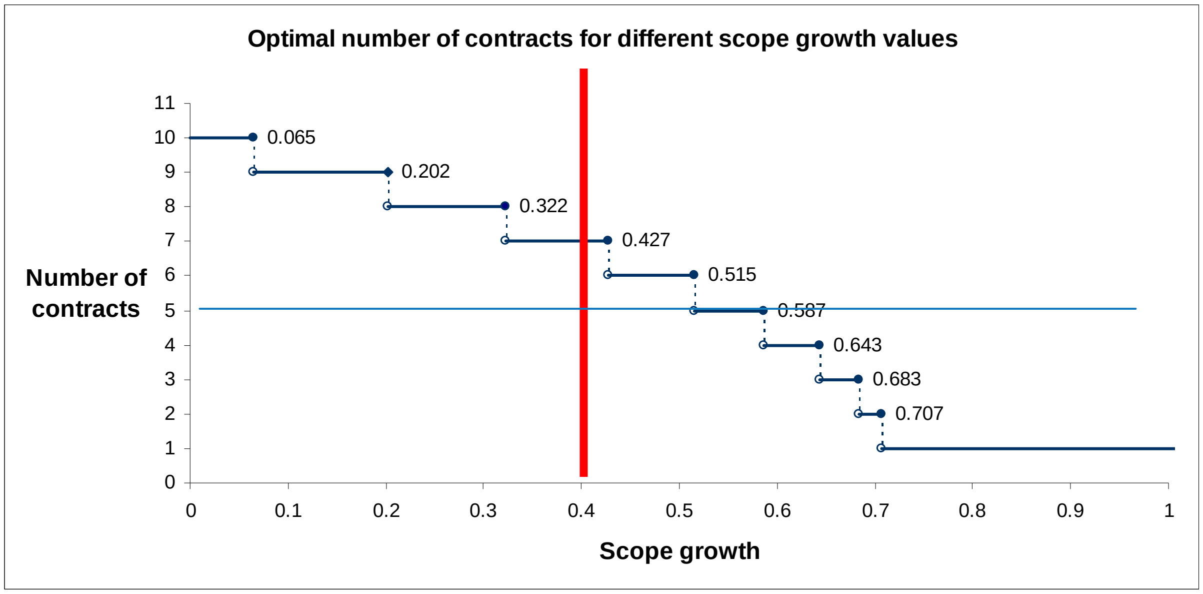


In figure 11, suppose the technical maturity for the program causes the decision-maker to assess there is enough risk to justify an expectation of scope growth in the program of 0.4 (40 percent). This is shown by the vertical red line and is a kind of demand curve for the decision-maker.

Also suppose the decision-maker is limited by personnel constraints to build a staff no larger than one required to manage 8 contracts. This is a kind of supply curve for the decision-maker. In this case the staffing constraint does not bind and the manager can choose to optimize his staff to manage 7 contracts. In this case our model suggests the ideal span of control is in fact 7.

Now let me modify my example. It is illustrated in figure 12 below.

Figure 12. Complexity and span of control: example 2



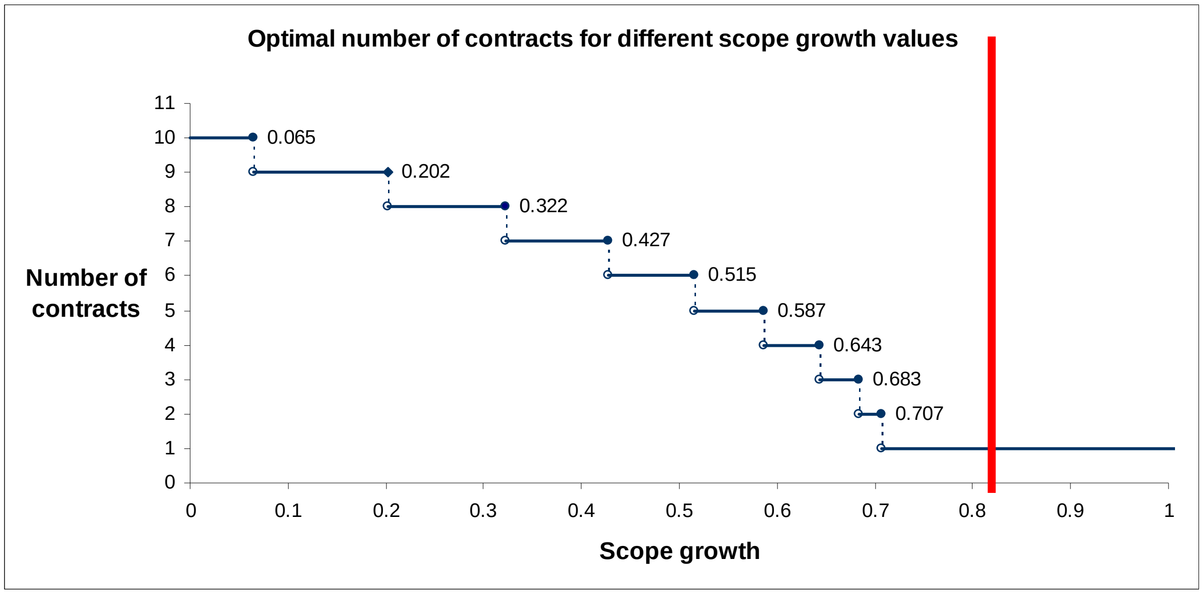
In this modified example the decision-maker still expects a scope growth of 0.4 (40 percent). However this time personnel constraints compel the decision-maker to build a staff no larger than one necessary to manage 5 contracts. Now the staffing constraint binds. The decision-maker would ideally use 7 contracts but will choose 5 contracts because of his staffing constraint. In this case the "best" span of control, all things considered, is 5.

So, my model provides empirical support for a methodology to choose the best span of control for a particular program. Furthermore, it shows how multiple answers to the span-of-control issue may be developed and justified.

Lead System Integrator (LSI)

Another issue that is related to the span of control issue is whether the government should integrate the components of a program itself or whether it should rely on a contractor (a prime contractor or alternatively a contractor acting as an LSI) to integrate the activities of the components of a program. I will use the diagram in figure 13 below to illustrate how my model might help answer this question.

Figure 13. Lead System Integrator



Suppose the level of technical maturity in an R&D program is very low. Suppose as a result a decision-maker expects a scope growth of 0.82 (82 percent). This is illustrated by the red vertical line in the above figure 13. In this case, my model suggests that the optimal number of contracts is 1. Hence, the government should ideally choose to write a contract with a single contractor (either a prime or an LSI).

In general, my model and the empirical evidence suggest an answer to the LSI issue. If the expectation of scope growth is very high due to low technical maturity, then go with a single prime contractor or with an LSI. On the other hand, if the expectation of scope growth is low due to high technical maturity, then use my previous span-of-control analysis to select the best number of contracts to manage the program (the best span of control). My analysis and the empirical evidence suggest that in no case should the number of contracts used to manage a program exceed 10. Beyond 10, the cost of the increased complexity is just too high and leads to total program cost growth that is higher than it should be.

Recommendations

In light of my analysis and the results I observed, I recommend the Navy use the simulation I developed to incorporate the effects of complexity into the initial estimation of R&D program costs.

The results of this research have certainly been encouraging. The results provide solid evidence for believing that program complexity contributes to cost growth. To improve the research and expand its application, I recommend that as much data as is available on TRLs be made available to us to demonstrate empirically the conjectured link between TRLs (initial technical maturity) and subsequent scope growth. Ideally, TRL data from a sufficiently large and robust sample size of completed programs would establish that link and allow us to model the exact nature of the relationship. This in turn could feed the simulation I developed to assess potential cost growth risk better.

In addition we could participate in efforts to define IRLs and the SRL better and to help develop metrics. If we could begin to collect data on these metrics, we could eventually incorporate these into our model of complexity and cost growth as well.

Finally, we could use the additional data and the analysis of results to improve current simulations and potentially construct new ones. Such efforts have the potential to significantly improve the realism of initial R&D program cost estimates. This could arm the decision-maker with better information to make better and timelier decisions.

This page intentionally left blank.

Conclusion

The ultimate purpose of this research project was to begin to explore the relationship between complexity and R&D program costs. I was able to get relevant data on 176 completed R&D programs to analyze.

The working hypothesis was that program complexity was a driver of R&D program cost growth. If this turned out to be true, an additional requirement was to see if there was a way to incorporate this relationship between complexity and cost growth into a method for improving the realism of initial program cost estimates at or before MS B.

The results of my analysis of a multiple regression model confirmed that indeed complexity was a predictor of program cost growth. The results were statistically significant. Moreover, I was able to model the specific functional relationship between the independent variables—complexity, scope growth, and cost overrun—and the dependent variable—program development cost growth.

As a result of this established relationship, I was able to demonstrate the feasibility of three applications of my model for improving cost estimates. I developed a simulation based on my model to paint a more nuanced picture of likely realistic program costs.

I also developed a method to determine an optimal span of control. This is the first time empirical evidence has been shown to support commonly held management beliefs about span of control. Third, I also developed a method to determine when the government should optimally use a single prime contractor or a contractor acting as an LSI.

In the penultimate section of the paper, I suggested avenues for further research. These suggestions include a rough outline of data requirements (at the very least TRLs on a robust and suitably sized sample set of completed R&D programs) and an outline of potential applications.

The exploration of the issue of program complexity is exciting research. Working on our own or in tandem with other agencies to perform continued research would have the happy advantage of being both very interesting and potentially very useful to future decision-makers.

Appendix A: Mathematics of the model

Let *scg* signify the independent variable of scope growth. Let *e* signify the independent variable of complexity measured as the number of edges plus one. Let *m* be an indicator variable to measure cost overrun. It takes a value of 1 if there has been at least 1 OTB in the program. It takes a value of 0 if there have been no OTBs. Let *k* represent a constant (intercept) term. Let β signify the coefficients associated with all the variables in the model. Let ε be a random error term that represents unobserved factors affecting program cost growth. The standard assumption is that $\varepsilon \sim N(0, \sigma^2)$. Finally let *cg* represent the dependent variable of R&D program cost growth. Then the model is:

$$cg = \beta_0 + \beta_1 \cdot scg + \beta_2 \cdot m + \beta_3 \cdot e + \beta_4 \cdot e^2 + \beta_5 \cdot scg \cdot e + \varepsilon$$

This page intentionally left blank.

Appendix B: Mathematics of elasticities

Let e_1 represent the complexity at point 1. Let e_2 represent the complexity at point 2. Let cg_1 be the cost growth at point 1 and cg_2 represent the cost growth at point 2. Let scg_1 be the scope growth at point 1. Let scg_2 be the scope growth at point 2. The formulae for arc elasticities are then:

- Elasticity of cost growth with respect to complexity:

  $\Delta cg(e_1 + e_2) / \Delta e(cg_1 + cg_2)$

- Elasticity of cost growth with respect to scope growth:

  $\Delta cg(scg_1 + scg_2) / \Delta scg(cg_1 + cg_2)$

Where cg_{med} is the median cost growth for the sample population, the formula for the point semi-elasticity of cost growth with respect to cost overrun is: $(\partial cg / \partial m)(1 / cg_{med})$

This page intentionally left blank.

Appendix C: Data tables

Table 6. Data for complexity analysis

Program	Program cost growth	Program scope growth	Indicator for overtarget baseline	Complexity	Complexity^2	Interaction of scope growth and complexity (scope growth times complexity)
1	1.76	0.48	1	1	1	0.48
2	0.38	0.31	0	7	49	2.17
3	0.55	0.34	0	1	1	0.34
4	1.16	1.09	0	16	256	17.47
5	0.32	0.17	0	1	1	0.17
6	1.05	0.61	0	7	49	4.26
7	0.64	0.64	0	1	1	0.64
8	0.89	0.81	0	16	256	12.94
9	0.04	0.04	0	2	4	0.08
10	0.31	0.17	0	1	1	0.17
11	0.81	0.55	0	1	1	0.55
12	0.52	0.26	0	2	4	0.53
13	2.41	2.33	0	1	1	2.33
14	0.09	0.05	0	37	1369	1.98
15	0.33	0.27	0	22	484	5.99
16	0.55	0.12	1	2	4	0.24
17	0.51	0.30	0	1	1	0.30
18	0.33	0.21	1	37	1369	7.77
19	0.37	0.12	0	2	4	0.24
20	0.23	0.04	1	16	256	0.70
21	1.26	1.00	0	4	16	4.01
22	0.70	0.70	0	2	4	1.40
23	0.30	0.10	0	4	16	0.40
24	0.94	0.52	0	1	1	0.52
25	1.11	1.02	0	1	1	1.02
26	0.18	0.06	0	1	1	0.06
27	0.74	0.41	1	7	49	2.85
28	0.06	0.02	0	4	16	0.10
29	0.57	0.20	0	4	16	0.79
30	0.37	0.20	0	7	49	1.42
31	0.57	0.45	0	4	16	1.80
32	0.31	0.07	0	11	121	0.82
33	0.61	0.15	1	2	4	0.31
34	0.78	0.41	0	16	256	6.61
35	0.76	0.00	0	1	1	0.00
36	0.07	0.00	0	1	1	0.00
37	0.05	0.01	0	2	4	0.01
38	0.24	0.24	0	11	121	2.68
39	0.35	0.35	0	1	1	0.35
40	0.55	0.82	0	1	1	0.82
41	0.08	0.06	0	7	49	0.44
42	0.31	0.08	0	1	1	0.08
43	0.09	0.07	1	67	4489	4.85
44	0.54	0.18	1	4	16	0.73
45	0.50	0.14	1	56	3136	8.00
46	0.34	0.45	0	2	4	0.89
47	6.80	6.28	0	1	1	6.28
48	1.04	0.82	0	2	4	1.65
49	0.34	0.21	0	16	256	3.32
50	0.73	0.63	0	2	4	1.26

Table 6. Data for complexity analysis, continued

Program	Program cost growth	Program scope growth	Indicator for overtarget baseline	Complexity	Complexity^2	Interaction of scope growth and complexity (scope growth times complexity)
51	1.28	0.74	1	7	49	5.21
52	0.10	0.72	0	2	4	1.45
53	0.05	0.03	0	7	49	0.20
54	0.33	0.09	1	11	121	0.97
55	3.28	1.97	0	22	484	43.26
56	-0.31	-0.39	0	2	4	-0.78
57	1.02	0.60	1	16	256	9.61
58	6.89	4.54	1	7	49	31.76
59	0.20	0.06	0	4	16	0.23
60	2.10	1.48	1	16	256	23.73
61	0.46	0.46	0	2	4	0.92
62	0.56	0.56	0	1	1	0.56
63	1.58	1.58	0	7	49	11.04
64	0.65	0.03	1	4	16	0.12
65	0.53	0.27	0	4	16	1.08
66	1.18	0.67	1	11	121	7.41
67	0.86	0.91	0	2	4	1.81
68	2.77	2.67	0	11	121	29.35
69	0.83	0.47	0	4	16	1.89
70	2.61	2.55	1	1	1	2.55
71	0.73	0.10	0	7	49	0.68
72	0.09	0.09	0	1	1	0.09
73	0.09	0.09	0	1	1	0.09
74	0.13	0.13	0	2	4	0.27
75	0.02	-0.01	0	7	49	-0.07
76	0.37	0.00	1	11	121	0.02
77	2.56	1.86	1	4	16	7.43
78	0.97	0.93	0	11	121	10.20
79	0.48	0.18	1	7	49	1.29
80	0.32	0.17	1	7	49	1.17
81	0.63	0.48	0	1	1	0.48
82	0.25	0.18	1	7	49	1.28
83	0.73	0.19	0	4	16	0.77
84	0.50	0.37	0	1	1	0.37
85	-0.01	0.05	0	1	1	0.05
86	-0.88	-0.89	0	11	121	-9.77
87	0.79	0.51	1	7	49	3.55
88	0.72	0.03	1	4	16	0.14
89	0.04	0.04	0	11	121	0.43
90	0.25	0.05	0	1	1	0.05
91	1.92	0.18	1	1	1	0.18
92	0.02	-0.01	0	4	16	-0.03
93	2.55	2.22	0	1	1	2.22
94	3.44	3.44	0	1	1	3.44
95	0.64	0.61	1	7	49	4.25
96	0.37	0.11	0	2	4	0.21
97	0.59	0.59	0	11	121	6.49
98	-0.06	-0.03	0	2	4	-0.06
99	0.92	0.60	0	4	16	2.39
100	2.06	1.96	1	2	4	3.93

Table 6. Data for complexity analysis, continued

Program	Program cost growth	Program scope growth	Indicator for overtarget baseline	Complexity	Complexity^2	Interaction of scope growth and complexity (scope growth times complexity)
101	0.45	0.20	0	2	4	0.39
102	0.32	0.29	0	7	49	2.00
103	1.22	0.74	0	7	49	5.17
104	2.74	2.07	0	1	1	2.07
105	1.14	0.02	1	1	1	0.02
106	0.03	0.00	1	7	49	0.03
107	0.20	0.02	0	1	1	0.02
108	0.00	0.00	0	1	1	0.00
109	0.28	0.28	0	2	4	0.56
110	0.15	0.16	0	16	256	2.48
111	0.03	0.00	0	1	1	0.00
112	0.08	0.06	0	1	1	0.06
113	4.60	4.39	1	1	1	4.39
114	0.17	0.04	1	1	1	0.04
115	0.07	0.00	0	1	1	0.00
116	1.52	1.48	1	2	4	2.97
117	0.08	0.04	0	1	1	0.04
118	0.22	0.00	0	2	4	0.00
119	0.37	0.66	0	1	1	0.66
120	0.36	0.20	0	7	49	1.37
121	0.07	0.04	0	1	1	0.04
122	0.12	0.01	0	2	4	0.03
123	0.10	0.06	0	1	1	0.06
124	0.00	0.00	0	1	1	0.00
125	0.36	0.36	0	16	256	5.79
126	0.88	0.02	0	1	1	0.02
127	0.37	0.23	0	7	49	1.59
128	0.74	0.74	0	1	1	0.74
129	0.52	0.08	1	1	1	0.08
130	0.30	0.28	0	29	841	8.08
131	0.70	0.44	0	2	4	0.88
132	0.40	0.29	0	4	16	1.15
133	0.55	0.33	0	7	49	2.31
134	-0.03	-0.03	0	2	4	-0.06
135	0.00	0.00	0	2	4	0.00
136	0.54	0.15	1	67	4489	9.77
137	0.67	0.48	0	2	4	0.95
138	0.02	0.01	0	2	4	0.01
139	0.47	0.25	1	7	49	1.74
140	0.49	0.28	0	7	49	1.97
141	0.55	0.51	0	11	121	5.66
142	0.09	0.03	0	1	1	0.03
143	0.59	0.01	0	2	4	0.02
144	0.12	0.12	0	2	4	0.24
145	1.74	0.03	0	1	1	0.03
146	0.58	0.04	0	1	1	0.04
147	0.25	0.09	0	1	1	0.09
148	0.25	0.19	0	56	3136	10.79
149	0.49	0.46	0	7	49	3.21
150	1.19	1.11	0	7	49	7.76

Table 6. Data for complexity analysis, continued

Program	Program cost growth	Program scope growth	Indicator for overtarget baseline	Complexity	Complexity^2	Interaction of scope growth and complexity (scope growth times complexity)
151	0.03	-0.09	0	2	4	-0.18
152	0.31	0.10	1	11	121	1.08
153	0.63	0.34	0	7	49	2.38
154	1.10	1.07	0	2	4	2.15
155	0.85	0.96	1	7	49	6.75
156	0.16	0.03	0	2	4	0.05
157	0.41	0.22	0	2	4	0.44
158	1.02	0.00	1	1	1	0.00
159	0.12	0.00	0	2	4	0.00
160	0.40	0.27	1	2	4	0.53
161	0.31	0.03	0	1	1	0.03
162	0.00	0.00	0	1	1	0.00
163	1.21	1.07	0	2	4	2.14
164	0.83	0.59	0	7	49	4.12
165	0.42	0.38	0	7	49	2.68
166	0.22	0.13	0	2	4	0.27
167	0.49	0.03	0	1	1	0.03
168	0.36	0.24	1	11	121	2.66
169	0.25	0.25	0	11	121	2.77
170	0.01	0.06	1	1	1	0.06
171	0.00	0.00	0	1	1	0.00
172	0.00	0.00	0	1	1	0.00
173	0.32	0.32	0	1	1	0.32
174	0.25	0.24	0	1	1	0.24
175	0.45	0.35	0	1	1	0.35
176	0.24	0.24	0	2	4	0.47

Glossary

CAS	Contract Analysis System
CBB	contract budget base
DoD	Department of Defense
R&D	research and development
i.i.d.	independent and identically distributed
LSI	lead system integrator
MS B	Milestone B
IRL	integration readiness level
OTB	over-target baseline
PM	project manager
SRL	system readiness level
TAB	total allocated budget
TRL	technical readiness level

This page intentionally left blank.

References

[1] Robert K. Garrett, Jr., Steve Anderson, Neil Baron, and James D. Moreland, Jr. *Managing the Interstitials, a System of Systems Framework Suited for the Ballistic Missle Defense System*, July 2009 (Research memorandum from the Naval Surface Warfare Center, Dahlgren Division, Dahlgren, VA)

[2] Gary Christle et.al. *Program Cost Growth: The Navy's Experience 1983-2003,* August 2004 (CNA Annotated Brief D0010181.A4/1Rev)

[3] M. Mitchell Waldrop. *Complexity*. New York: Simon & Schuster, 1992

[4] Edward N. Lorenz. *The Essence of Chaos*. Seattle: University of Washington Press, 1993

[5] Steven Strogatz. *Sync*. New York: Hyperion, 2003

[6] David R. Anderson, Dennis J. Sweeney, and Thomas A. Williams. *Statistics for Business and Economics,* 9th edition. Cincinnati: South-Western College Publishing, 2005

[7] Herbert A. Simon. *Administrative Behavior*. New York: The Free Press, 1997

[8] R. H. Coase. *The Firm the Market and the Law*. Chicago: University of Chicago Press, 1990

[9] Oliver E. Williamson. *The Mechanism of Governance*. Oxford: Oxford University Press, 1996

This Page intentionally let blank.

List of figures

This Page intentionally let blank.

List of tables

This page intentionally left blank.

CRM D0022312.A2/Final