Software Engineering Institute
Carnegie Mellon University

THE QUELCE METHOD: USING CHANGE DRIVERS TO ESTIMATE PROGRAM COSTS

Sarah Sheard

April 2016

Abstract

Problems with cost estimation, ranging from estimator overconfidence to unintegrated tools, result in potentially billions of dollars of unanticipated expenses for DoD programs. Quantifying Uncertainty in Early Lifecycle Cost Estimation (QUELCE) is a method, developed by the Carnegie Mellon University Software Engineering Institute, for estimating potential program costs in a way that acknowledges and uses uncertainty that occurs early in the development lifecycle. This report first familiarizes the reader with the QUELCE method. QUELCE computes a distribution of program costs based on Monte Carlo analysis of program cost drivers—assessed via analyses of dependency structure matrices and Bayesian belief networks—and a standard project estimation tool. The analyses are based on change drivers, or changes that might occur that would substantially change the cost outcome of a program. The report then provides the current organization scheme of change drivers and describes how each one is used to determine any additional impacts that should be folded into the cost estimate. Finally, it introduces elaborations to the change drivers for application to sustainment-phase programs.

Introduction: The Cost Estimation Challenge

Because large-scale programs such as Major Defense Acquisition Programs (MDAPs) and Major Automated Information Systems (MAIS) cost billions of dollars [GAO 2009], incorrect cost estimates may lead to enormously costly mistaken decisions. Poor cost estimates in Department of Defense (DoD) programs cause cost overruns, significant schedule delays, and compromised technical capability [Bolton 2008, Hofbauer 2011].

In 2012, the Office of Performance Assessments and Root Cause Analyses (PARCA) performed root cause analyses of program cost overruns in the DoD and identified unrealistic cost and schedule estimates as major contributors to cost growth. PARCA concluded that unrealistic estimates are generally caused by invalid major assumptions, rather than by methodological errors. These “framing assumptions” are made at the start of a program and become so embedded in the thought process that they are never challenged [Bliss 2012].

Improvements in cost estimation that would make these assumptions more precise and reduce early lifecycle uncertainty can save potentially billions of dollars by allowing accurate program funding that is enough to permit program success without being wasteful. However, early cost estimates are difficult

to get right. Domain experts, cost analysts, and program managers must make judgments about software-reliant systems that are very complex and often unprecedented. Estimation practice typically has the following characteristics:

- In the early lifecycle, a known uncertainty about parameters needed for the estimate makes it difficult to use historical data.
- While the uncertainties about many parameters are known, there is no systematic method for combining uncertainties or understanding the relationships among them.
- Experts are often overconfident about their judgments.
- Often the only cost estimation scenario that is explicitly chosen is a single planned case. The assumptions that underlie the scenario are not made explicit. While the Analysis of Alternatives phase defines and assesses multiple scenarios, they are not necessarily chosen in a systematic manner, and their assumptions may not be explicit.
- Even when events that may invalidate the estimate are predicted, updating the estimates with new knowledge occurs, at best, periodically.
- A single estimate is known to be wrong, but there is no way to calculate how wrong it might be. For example, the Weapons System Acquisition Reform Act of 2009 required estimates that had an 80% level of confidence, but two years later the specific confidence level was removed since there was no good way to calculate it [PL 111-23, PL 111-383].
- Cost estimation tools are not integrated; much work must be done to ensure that the outputs of one tool are properly recalculated to serve as inputs to the next tool.

Since 2010, the Carnegie Mellon University Software Engineering Institute (SEI) has done research and analysis to address these very factors. The result is a method called Quantifying Uncertainty in Early Lifecycle Cost Estimation (QUELCE).

The purpose of this report is threefold:

- to give the reader a clear high-level overview about the QUELCE method, including what it needs as inputs, what it produces as outputs, and what steps it takes to produce the outputs
- to expose our current list of change drivers, or changes that might occur that would substantially change the cost of a program, which we have evolved first through analysis and then by application to many programs, and show how they can be used to produce better estimations of cost
- to introduce change driver elaborations that apply to sustainment-phase programs rather than to the early lifecycle

The QUELCE Method

The first purpose of this report is to provide an overview of QUELCE to readers mostly likely to benefit from applying the method, including senior-level MDAP and MAIS personnel, estimators and analysts, and subject-matter experts. The QUELCE method quantifies uncertainty early in a program's lifecycle and produces cost estimates with ranges and distributions for specific scenarios. With QUELCE outputs, managers can scope a program more accurately, thereby improving worst-case estimates of program

size, defects, cost, duration, and other values. Additionally, QUELCE can be used to estimate costs for a number of alternatives, thus allowing a more evidence-based tradeoff among capabilities.

The QUELCE method improves the standard way of creating program cost estimates in several ways.

1. QUELCE provides a structured way of developing cost estimates that builds on historical information and expert judgment.
2. QUELCE trains experts in judgment, enabling them to determine how well they know what they think is true and how wide to make their 90% confidence interval, given the degree of correctness of previous estimates.
3. QUELCE defines typical change drivers based on historical experience with many programs, asks the trained experts to estimate how likely these and other change drivers are to occur, and asks them how much program change the change drivers would cause if they do occur.
4. QUELCE identifies and evaluates both marginal and conditional probabilities for the changes that could occur.
5. QUELCE uses a combination of sophisticated tools to estimate a probability distribution of costs for a large number of possible scenarios. These include
   - a dependency structure matrix to understand the interaction of change drivers for a specific project
   - a Bayesian belief network to model how those interactions affect outcomes for a given scenario
   - cost estimating relationships from popular software estimation tools to evaluate cost given effort and other inputs
   - Monte Carlo analysis to produce the range and confidence levels of the estimates
6. Instead of a single likely estimate, QUELCE produces a distribution of cost estimates, showing the most likely cost and the likelihoods of lesser and greater costs.

The steps of the QUELCE method can be performed internally by an organization if its analysts and others are knowledgeable enough. However, most organizations choose to employ consultants from the SEI for several steps. This has the benefit of raising the knowledge level of the cost-estimation community as well, since SEI consultants can fold the lessons back into the QUELCE documentation.

The major steps of QUELCE are as follows and are summarized in Figure 1.

The QUELCE Workshop

A workshop pulls together experts in the program and the domain to understand what is known and what is uncertain for the program in question. It includes an expert-judgment calibration module, in which a QUELCE facilitator asks experts to estimate the answers to general-knowledge questions such as the life span of a wild pigeon or the height of the Washington Monument. They are prompted to give their answers as a range that is wide enough that there is a 90% chance that the actual answer falls within the range. Their estimates are checked to see how well they predicted the actual number. Most experts overestimate how good their guesses are, so fewer than 90% of their predictions turn out to be accurate. Experts then try again with different questions until they truly can predict 90% of the numbers when aiming for a 90% confidence level.

Elicit Change Drivers and Alternatives

During the main portion of the workshop, experts and program staff review a standard set of change drivers and determine how well they apply to the program being estimated. They identify the extent to which one change driver can influence other change drivers: for example, a change in stakeholders will create new requirements. They also identify the states that each change driver may take—such as no change, minor change, and major change—and the likelihood of each of those states. They estimate the size of the proposed software and define an initial set of scenarios (a particular instance of some changes happening and others not happening).

Reduce Complexity with Cause-and-Effect Analysis

Analysis for the next step is done by the SEI or by company analysts. In this step, analysts create a dependency structure matrix (DSM) of the change drivers. Next, they attempt to make the connections in the DSM appear only in one triangular half-matrix by reordering the rows and columns. With a triangular matrix, the analysts can construct the Bayesian belief network (BBN) used to analyze the probabilities. They look for cycles, representing feedback loops in the DSM; in a cycle, one change driver both influences a second change driver and is influenced by it (possibly through a roundabout path involving other change drivers). Cycles are detected when values cannot be moved into a triangular half-matrix.

Removing cycles is one way to reduce the complexity of the DSM and therefore enable creation of a triangular half-matrix. Sometimes a cycle can be resolved by decomposing a change driver into two more granular change drivers, usually based on the natural time sequence in which the changes would occur. Other times, combining two strongly interrelated change drivers can resolve the cycle. Another way to reduce the complexity is to remove some of the less important dependencies among change drivers.

The matrix reduction process uses an automated tool that quickly identifies the next cyclic dependency. At any given point in the process, the tool provides a complete visualization of the graph.

Assign Conditional Probabilities

The resulting directed, acyclic graph is the basis for constructing the BBN. The BBN graph displays change drivers as nodes; arrows between nodes show how one change driver affects another (impact of an upstream node on a downstream node). Impact likelihoods are calculated with conditional probability equations, whose inputs were elicited in the workshop. For example, if Node C is influenced by (preceded by) Node A and Node B, the conditional probability for Node C might be calculated as $CP(C) = 0.35 + 0.042\ CP(A) + 0.22\ CP(B)$.

To identify under what conditions the program will calculate estimates, experts identify scenarios. To create a scenario, the analysts choose a state for one or more change drivers. There is always a nominal scenario, which is when all change drivers are in the no-change state. Other scenarios examine what happens when some of the change drivers are set to non-nominal states. For example, suppose Node A influences Node B, which influences Node C. If changing stakeholders is Node A, and changing requirements is Node C, then one scenario might assume that stakeholders have changed (set Node A to “changes occur”) and calculate Node C for all possible states of Node B.

Later, a program can adapt the initial scenarios developed in the workshop for new knowledge, such as to incorporate additional stakeholders' concerns.

Apply Uncertainty to Cost Formula Inputs for Scenarios

In the next step of the QUELCE method, analysts identify how the BBN relates to the selected cost estimation tool inputs. Various cost estimating tools—such as COCOMO, CoSYSMo, and SEER—estimate cost based on different inputs. We use "glue" nodes to connect the BBN to the various cost estimation models that the program uses. Glue nodes are inserted into the BBN to calculate outputs that the chosen cost estimation tool then uses as inputs, such as the expected size of the software to be built.

Perform Monte Carlo Simulation to Compute Cost Distribution

We use a combination of an Excel spreadsheet and a Monte Carlo tool called Oracle Crystal Ball to create probability distributions for program cost. Monte Carlo techniques run various scenarios a large number of times to get a distribution of outputs for each scenario. For each scenario, the analyst chooses the states of some change drivers, and Crystal Ball chooses the remaining change driver states randomly. This gives a single set of inputs to the cost estimation tool and produces a single output cost. The process of randomizing the inputs and getting a single cost output is repeated over and over. These multiple runs produce a distribution of cost outputs for the scenario. A program can identify other scenarios and calculate their cost distributions in the same way.

Estimate Program Cost More Accurately

Decision makers should review the output cost distributions to determine the program cost estimates that they will use in their budgeting process. They can ask the analysts to run additional scenarios. For example, they can imagine mitigating one of the risks, thus setting some of the driver states to nominal, and see how that would affect the cost budget.

Figure 1 shows an overview of the QUELCE method and illustrates the steps that create the products. The workshop participants create the driver state table, perform the cause-and-effect analysis that determines which change drivers affect which others, and consider the likelihood of each change (Steps 1 and 2 in the figure). Workshop participants may also suggest the scenarios used in Step 3. Offline analysis reduces the cycles and complexity of the DSM and creates the BBN from it (Steps 2 and 3), determines how to tie BBN model outputs to the desired cost estimation tool (Step 4), and develops the cost distribution using the Monte Carlo simulation (Step 5).

Figure 2 shows the flow of analysis; the step numbers are the same as in Figure 1. The workshop performs most of the tasks for Steps 1 and 2 and determines initial scenarios and a size estimate. Offline analysis performs most of the tasks for Step 3 and the analysis in Steps 4 and 5.

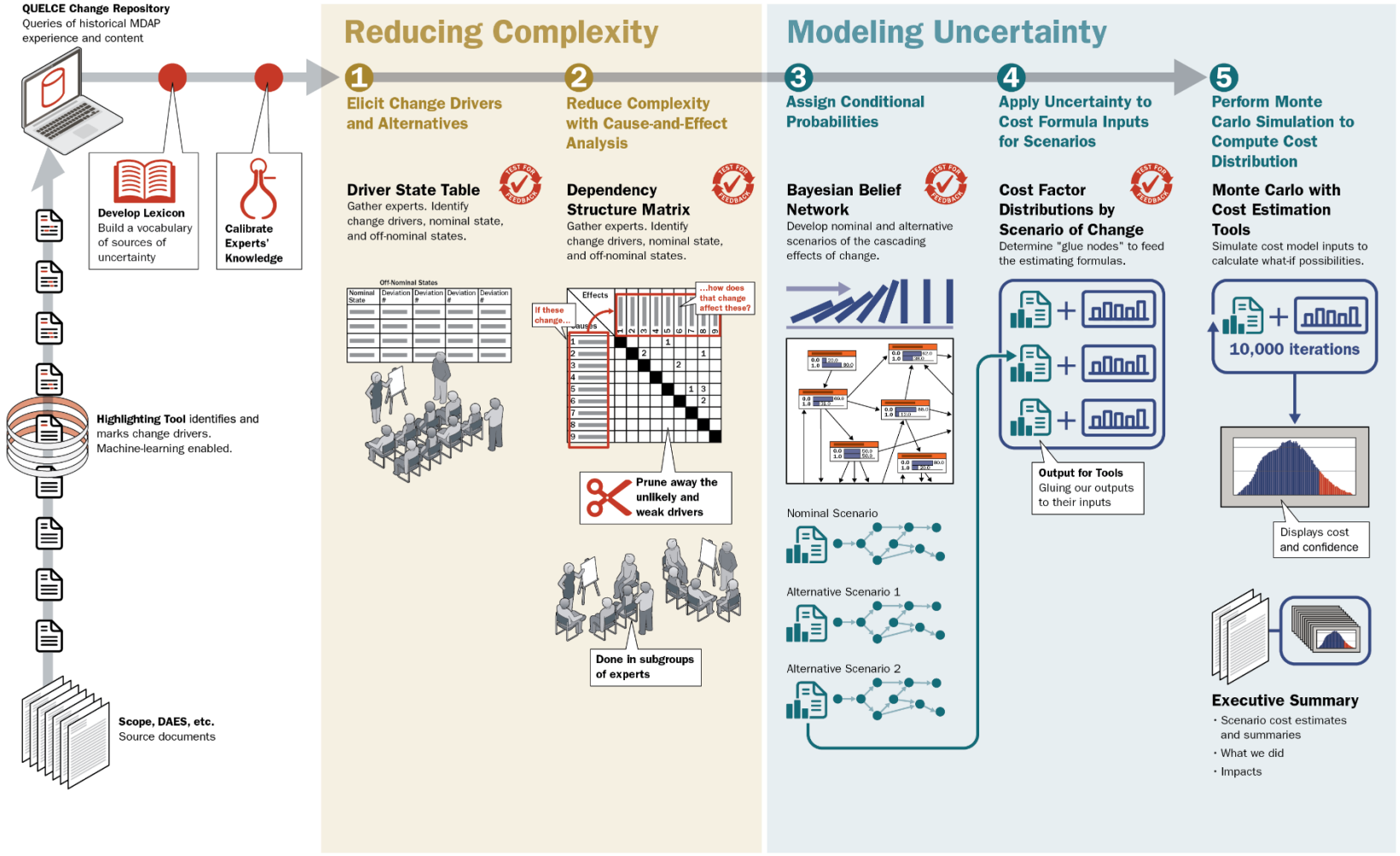

Figure 1: QUELCE Products

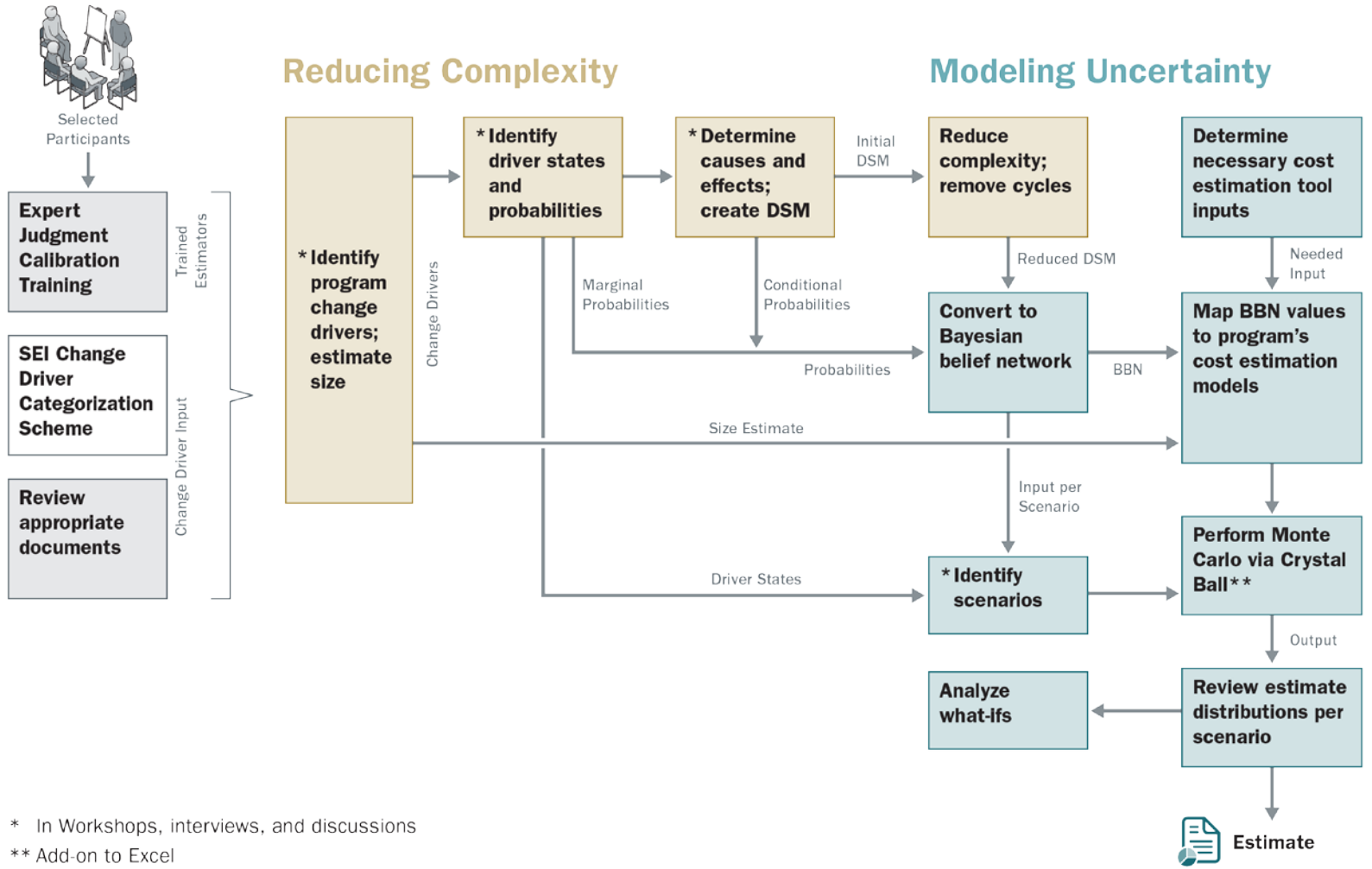

Figure 2: QUELCE Analysis Flow

Change Drivers: Overview

The second purpose of this report is to document the SEI's knowledge about typical program change drivers. Change drivers are possible conditions or events that were not accounted for in a program's baseline estimate; thus, if their state becomes other than nominal, a change in program cost, schedule, or performance will result. In this document, we consider only the effects of change drivers on cost.

To help programs identify these change drivers, the SEI has developed a repository of change drivers that incorporates several years of experience preparing and sharing the QUELCE method. The SEI repository of change drivers has evolved over time. We collected the first set from literature, used it to examine a few initial programs, and then used those experiences to revise the set of change drivers. We later recategorized the change drivers to facilitate a reproducible review of program documents. Our current classification categories are shown in Table 1. Descriptions of the change drivers are provided in the next section.

Note that these categorizations are groupings. Programs will still want to talk through different aspects of a development effort implied by the categorizations and identify specific change drivers that may impact their program. This is one of the focus areas of the QUELCE Workshop.

Early assumptions are best documented by the subject-matter experts (SMEs) who are able to conceptualize alternatives. Change drivers may be recognized in assumptions about maturity of technology, capability of the engineering process, stability of stakeholder concerns, and program management factors such as funding and program organization. Thus, successful execution of QUELCE requires that SMEs from these domains are available to document their assumptions and possible alternatives.

Examples:

- A new stakeholder joins a program late, or an existing stakeholder withdraws support.
- A technology at Technology Readiness Level (TRL) 4 is selected for design at Milestone A, but the technology does not achieve TRL 7 by Milestone C, so the program must use an alternative technology, lengthen the schedule until the technology works in production, or descope the product to omit the required technical capability.
- A selected contractor is unable to develop a component and must be replaced.
- Funding is cut by 20%, causing significant replanning of the program.

Each of these examples will cause many different undesired effects. The QUELCE method creates a record of the original assumed states and the resulting undesired effects of each change.

Change Drivers for Early Programs and for Sustainment

The QUELCE method was first used to focus on the early lifecycle. More recently, the SEI interviewed several sustainment organizations about whether the same change drivers apply to programs undergoing software sustainment. Because software sustainment necessarily involves making changes in the software, it also necessarily involves software engineering. Therefore, all change drivers in Table 1 apply

to sustainment as well as to the early lifecycle. In some cases, we discuss the differences between the applications of the change driver in the two phases.

Table 2 lists change drivers that are not in the early lifecycle list and apply more to sustainment than to new-start programs. We refer to these change drivers by number in the descriptions in the Elaborations for the Sustainment Phase section following the tables. More detail on change drivers elaborated for the sustainment phase is available there.

Software changes during sustainment always require engineering work. Such changes are commonly grouped into the following categories:

- Corrective: These changes result from imperfect operational performance, whether they are due to defects or the inability to meet a performance goal.
- Adaptive: These changes are the consequence of external changes, such as change to an interface, network, or database.
- Perfective: These are enhancements made to improve performance or add features that help the warfighter deliver additional value. They are also made to attract new uses of the product.
- Preventive: In software, this is primarily performed for cyber concerns—to protect against new threats or to remove newly discovered cyber vulnerabilities.

In addition to standard software changes, the following specific concerns also arise during software sustainment:

- Software license agreements can change as using organizations and licensing organizations evolve. Changes in software version may propagate to other components.
- The software experiences more external software changes such as changes to interfaces, data sources, and network management functions.
- Replacing sensors or other operational hardware with more capable (i.e., software-enabled) versions may change data and network interfaces.

Change Driver Categories for the Early Lifecycle

Table 1 gives the categories of the original set of change drivers for the early lifecycle.

Table 1: Categorized Change Drivers

A Acquisition Environment	
A1	Acquisition Category (ACAT) Status
A2	Governance, Policies, and Standards
A3	External Interdependencies/Coordination
A4	External Stakeholders
A5	External Events
A6	Other: Acquisition Environment
B Acquisition Management	
B1	Acquisition Strategy
B2	Contracting
B3	Management Structure
B4	Program Scope
B5	Budget

B6	Schedule
B7	Staffing
B8	Facilities, Support Technology, and Equipment
B9	Program Information Management
B10	Program-Contractor Performance
B11	Other: Acquisition Management
C. Engineering Solution/Work Products	
C1	Conceptual Design/Requirements
C2	System Architecture and Design
C3	Production and Construction
C4	Certification and Accreditation
C5	Deployment, Operations, and Support
C6	Technology Maturity/Readiness
C7	Estimated Complexity/Difficulty
C8	Supply Chain Products
C9	Other: Engineering Solution/Work Products

Change Driver Elaborations for the Sustainment Phase

In Table 2, change drivers from the categorization scheme in Table 1 are referenced by the first two digits of the elaboration number. For example, in Table 1, Category A3 is External Interdependencies/Coordination. This category has two elaborations for the sustainment phase in Table 2, numbered A3S1 and A3S2. The S emphasizes a sustainment-phase elaboration. Sometimes the same elaboration relates to multiple change drivers. For instance, "Contractor hands off to organic organization" fits with Category A3: External Interdependencies/Coordination and with B3: Management Structure, so it appears both as A3S1 and as B3S1.

Table 2: Change Driver Elaboration for the Sustainment Phase

A. Elaborations to Change Drivers for the Acquisition Environment	
A3S1	Program performers experience challenges learning lessons across badge colors (across the different organizations who manage people working together during the sustainment phase).
A3S2	Contractor hands off to organic organization for sustainment. Amount of knowledge transfer is a function of the time spent working on a program together.
A4S1	Relationships among the sustainment stakeholders degrade.
A5S1	Information assurance/cybersecurity surprises require redesign or retrofit of hardware, software, or both (HW/SW).
B. Elaborations to Change Drivers for Acquisition Management	
B3S1	Contractor hands off to organic organization for sustainment.
B5S1	Color-of-money problems delay sustainment tasks.
B7S1	Experience level of SW engineers is not sufficient.
B7S2	Program has difficulty hiring people.
B8S1	Disparate commercial tools need to be cobbled together.
B8S2	Program needs to rework facility environment.
B9S1	Government, contractors, and vendors argue over data rights.
B10S1	Contractor hands off to organic organization for sustainment.
B11S1	Contractor hands off to organic organization for sustainment.
B12	Information assurance/cybersecurity surprises require redesign or retrofit of HW/SW. (Note: We have added a change driver B12, rather than considering this to be just an elaboration.)

C. Elaborations to Change Drivers for Engineering Solution/Work Products	
C1S1	Information assurance/cybersecurity surprises require redesign or retrofit of HW/SW.
C1S2	Program experiences unfunded (or funded) capability creep; e.g., operational users change their minds.
C2S1	Information assurance/cybersecurity surprises require redesign or retrofit of HW/SW.
C5S1	System can no longer be sustained, so starts fresh (SW fragility).
C10	Information assurance/cybersecurity surprises require redesign or retrofit of HW/SW.

Categorized Change Drivers: Descriptions

The change driver categories separate change drivers into Acquisition Environment, Acquisition Management, and Engineering Solution/Work Products. This organization of change drivers was chosen to maximize the similarity in judgments of different domain experts. In workshops, when experts are selecting and assessing change drivers, the change driver categories should be intuitive so that the list of change drivers can converge quickly.

As part of the workshop, SMEs identify change drivers for each category that are specific to the program. These program-specific change drivers are used in the DSM and the BBN in the same way as standard change drivers.

A Acquisition Environment

Acquisition Environment change drivers are unanticipated conditions or events related to the context, circumstances, and setting in which an acquisition program must function. They are

A1 Acquisition Category (ACAT) Status

A2 Governance, Policies, and Standards

A3 External Interdependencies/Coordination

A4 External Stakeholders

A5 External Events

A6 Other: Acquisition Environment

A1 Acquisition Category (ACAT) Status

These change drivers are unanticipated conditions or events that relate to the ACAT status of the program (ACAT 1 programs are the biggest). Getting the correct ACAT status may facilitate program decision making and execution and help ensure compliance with statutorily imposed requirements. Acquisition categories determine the level of review, decision authority, and applicable procedures.

A2 Governance, Policies, and Standards

These change drivers are unanticipated conditions or events that could change the practices, policies, standards, laws, and regulations with which a program must comply.

A3 External Interdependencies/Coordination

These change drivers are unanticipated conditions or events that relate to the network of external organizations that contribute to or influence the program. Execution of program activities requires coordination among the involved organizations. Examples include interdependencies among programs and interdependencies among services or agencies.

A4 External Stakeholders

These change drivers are unanticipated conditions or events related to any individual or group considered to be outside of the program that can exert influence on the direction of the program or the resources available to the program. Examples of external stakeholders include the program executive office, military service, and Congress.

A5 External Events

These change drivers are unanticipated events such as natural disasters and other types of emergencies that confront organizations. Examples include floods, fire, disruption of utilities, environmental hazards, and civil disruption such as war, rioting, revolution, protests, vandalism, and terrorism.

A6 Other: Acquisition Environment

These change drivers are unanticipated conditions or events related to the program's environment that do not map to any other codes.

B Acquisition Management

Acquisition Management change drivers are unanticipated conditions or events related to the management function that coordinates the program office and contractors to accomplish goals and using available resources efficiently and effectively. Acquisition management includes planning, organizing, staffing, leading or directing, and controlling the program office and contractors to accomplish a goal. The change drivers in this category are

- B1 Acquisition Strategy
- B2 Contracting
- B3 Management Structure
- B4 Program Scope
- B5 Budget
- B6 Schedule
- B7 Staffing
- B8 Facilities, Support Technology, and Equipment
- B9 Program Information Management
- B10 Program-Contractor Performance
- B11 Other: Acquisition Management

Note that B12, a 12th change driver, is recommended for sustainment in the section Elaborations for the Sustainment Phase.

B1 Acquisition Strategy

These change drivers are unanticipated conditions or events related to the program's approach to acquiring systems, based on considerations of supply sources, acquisition methods, requirements specification types, agreement types, and related acquisition risks.

B2 Contracting

These change drivers are unanticipated conditions or events related to the program's process of setting up a mutually binding legal relationship obligating the seller to furnish systems (including construction) and the buyer to pay for them. Examples include supplier agreements, licenses, and memoranda of agreement.

B3 Management Structure

These change drivers are unanticipated conditions or events related to the hierarchy used to manage performance in an organization.

B4 Program Scope

These change drivers are unanticipated conditions or events related to the program's objectives, including the activities performed by the acquirer as well as the milestones and deliverables for suppliers.

B5 Budget

These change drivers are unanticipated conditions or events related to the program's budget, namely, the estimated funding available within a given period of time.

B6 Schedule

These change drivers are unanticipated conditions or events related to the program's scheduled tasks or activities that must be accomplished in a specific sequence within a given period of time.

B7 Staffing

These change drivers are unanticipated conditions or events related to the program's supply of program and contractor personnel (i.e., labor) available to perform all activities.

B8 Facilities, Support Technology, and Equipment

These change drivers are unanticipated conditions or events related to the program's supply of facilities, technologies, and equipment needed to support program and contractor activities.

B9 Program Information Management

These change drivers are unanticipated conditions or events related to the program's ability to identify, document, and manage its information needs, infrastructure support, and information technology assets. Examples include information sharing, information assurance, data interoperability, data ownership, and data dependencies.

B10 Program-Contractor Performance

These change drivers are unanticipated conditions or events related to the operation of carrying out program and contractor activities within budget and schedule parameters. Execution of program and contractor activities requires coordination among the involved organizations. Examples include program office performance, contractor performance, subcontractor performance, engineering performance, and supply-chain performance.

B11 Other: Acquisition Management

These change drivers are unanticipated conditions or events related to the acquisition management that do not map to any other codes.

C. Engineering Solution/Work Products

Engineering Solution/Work Products change drivers are unanticipated conditions or events related directly to the systems, software, and artifacts that the program will produce. These change drivers are listed in lifecycle order, from requirements to deployment, followed by change drivers that do not relate to a lifecycle phase. Note that there is no specific change driver for the sustainment phase because software sustainment work is also engineering, so many of the early phase change drivers also apply. Rather, elaborations of change drivers for the sustainment phase are covered separately in the following section.

The change drivers in this category are

- C1 Conceptual Design/Requirements
- C2 System Architecture and Design
- C3 Production and Construction
- C4 Certification and Accreditation
- C5 Deployment, Operations, and Support
- C6 Technology Maturity/Readiness
- C7 Estimated Complexity/Difficulty
- C8 Supply Chain Products
- C9 Other: Engineering Solution/Work Products

Note that C10, a 10th change driver, is recommended for sustainment in the section Elaborations for the Sustainment Phase.

C1 Conceptual Design/Requirements

These change drivers are unanticipated conditions or events related to the set of conditions and capabilities that must be met or possessed by a system. The conceptual design stage is where a program examines an identified need, defines requirements for potential solutions, evaluates potential solutions, and develops a system specification. The system specification represents the technical requirements that will provide overall guidance for system design.

C2 System Architecture and Design

These change drivers are unanticipated conditions or events related to the structure and associated behavior of a system. System architecture and design includes the development of detailed designs for the system. It also includes the design and specification of subsystems that perform the desired system functions in compliance with the system requirements.

C3 Production and Construction

These change drivers are unanticipated conditions or events related to building and testing a system. The product is built or assembled in accordance with the requirements and design. It is then tested in

the target operational environment. Key steps in this stage include production and construction of system components, coding of software, acceptance testing, system integration, and operational testing and evaluation.

C4 Certification and Accreditation
These change drivers are unanticipated conditions or events related to obtaining certificates and accreditations, such as by evaluating, describing, testing, and authorizing systems or activities before or after a system is in operation.

C5 Deployment, Operations, and Support
These change drivers are unanticipated conditions or events related to deploying the system, using the system in the intended operational setting, and maintaining the system over time. Also see Elaborations for the Sustainment Phase.

C6 Technology Maturity/Readiness
These change drivers are unanticipated conditions or events related to the extent to which technology to be used on the program is suitable for immediate application. A mature technology is one that has been in use for long enough that most of its initial faults and inherent problems have been removed or reduced by further development. TRL measures assess the maturity of evolving technologies (devices, materials, components, software, work processes, etc.) during their development and, in some cases, during early operations.

C7 Estimated Complexity/Difficulty
These change drivers are unanticipated conditions or events related to the extent to which a program task or activity is perceived to be complex or difficult. In many cases, tasks and activities become more complex or difficult to complete than originally anticipated.

C8 Supply Chain Products
These change drivers are unanticipated conditions or events related to the products and services provided by suppliers throughout the program's supply chain. A supply chain is a network of organizations, people, activities, information, and resources involved in moving a product or service from supplier to customer. Any issues or problems with products and services provided by suppliers can lead to quality problems and rework.

C9 Other: Engineering Solution/Work Products
These change drivers are unanticipated conditions or events related to the engineering solution and work products that do not map to any of the provided codes.

Elaborations for the Sustainment Phase: Descriptions

After SEI did considerable work with programs on quantifying uncertainty in early lifecycle cost estimation, clients also expressed interest in using the QUELCE method to help estimate the cost of sustainment.

Interviews with sustainers unearthed a number of new or somewhat different change drivers. We present the somewhat-different ones here as sustainment elaborations, tied to relevant change driver categories for the early lifecycle described in the previous section. Their category numbers show the change driver number first, followed by an S for sustainment, then a 1 or 2 for the elaboration related to that change driver.

Two new change drivers, B12 and C10, are included as well. They should be categorized under Acquisition Management and Engineering Solution/Work Products, respectively.

Elaborations of A: Acquisition Environment Categories for the Sustainment Phase

A3S1. Program performers experience challenges learning lessons across badge colors. This change driver acknowledges that processes are performed by a team consisting of employees from more than one organization. Such experiences occur in early phases as well, but in the sustainment phase, there are more different badges (various military, FFRDC, contractor, and subcontractor organizations) all working closely together, with only a small number of people with each specific badge. Early on, the organizations tend to have different tasks that need to be coordinated on an organization-to-organization basis; in the sustainment phase, it is individuals who have to coordinate, and often the relationships between people of different badges who are working on the same task are stronger than the relationships among people with the same badge who are working on different tasks.

A3S2. Contractor hands off to organic organization for sustainment. Amount of knowledge transfer is a function of the time spent working on a program together. This is a major change, not relevant to the early phases (except as there should be planning and accommodations for this eventually happening; those activities are noted separately under other categories). This change is extremely important in the sustainment phase; data rights transfer is a common example. The budget must be allocated to ensure that the organic personnel are well informed and all planning is accomplished thoroughly (see B9).

A4S1. Relationships among the variety of sustainment stakeholders degrade. This change driver is similar to the early phase change driver, with a few variations. In the sustainment phase, there are fewer people from each stakeholder organization working on the program. In addition, organizations tend to place less emphasis on programs in the sustainment phase, so it is not very satisfying or rewarding to work for long times on such programs. Also, the sustainment phase is ordinarily much longer than the original acquisition phase; thus, the potential for loss of organizational knowledge is higher.

A5S1. Information assurance/cybersecurity surprises require redesign or retrofit of HW/SW. This is one of several sustainment-phase elaborations that relate to security. The External Events change driver was originally intended for events like hurricanes, which are unpredicted and sudden. Here, the external event is that the world of security adversaries and threats evolves away from the situation during acquisition. The fact that threats will evolve is certain and ongoing, but the kind of evolution that will occur is unpredictable.

When threats change, software often has to change to keep up with them. Thus, this change driver also appears in C2 (redesign needed) and should appear as evolving requirements (perhaps a sustainment elaboration to the original statement of requirements, C1). Two new categories should be added as well. In Operational Management, there should be a B12 to address the administrative and organizational

aspects of the evolving security situation, such as updating policies and accesses as well as administering processes that prevent security problems. In Engineering Solution/Work Products, there should be a C10 to address the technical aspects of continually evolving security requirements during the sustainment phase.

Elaborations of B: Acquisition Management Categories for the Sustainment Phase

B3S1. Contractor hands off to organic organization for sustainment. The amount of knowledge transfer between organizations is a function of the time spent working on a program together. The management structure of the sustainment effort changes drastically if the responsibility for maintaining the software shifts from the original designers (the contractor) to an organic organization.

B5S1. Color-of-money problems delay sustainment tasks. This is theoretically no different from the problems in earlier time frames; however, the sustainment phase can magnify such problems because the majority of sustainment funding comes from O&M funding (3400 codes) rather than the RDT&E funding, which tends to occur in greater amounts (3600 codes), and various procurement funding (3010–3080 codes).

The funding problem is complicated by a general difference between software and hardware sustainment. Unlike hardware, software sustainment does *not* mean restoring the original condition of the item; rather, it means changing the item. Typical reasons are that the software environment changed (adaptive software maintenance); new or changed user requirements result in functional enhancements (perfective maintenance); bugs should be fixed (corrective maintenance); or software maintainability or reliability should be increased (preventive maintenance).

B7S1. Experience level of SW engineers is not sufficient. The needed experience level is higher than the experience level that can be hired.

B7S2. Program has difficulty hiring people. The people being sought tend not to want to live near a military base, accept the government pay scale, or deal with the slow speed of hiring. If they are hired, they might then wait up to a year for their security clearance before they can fully perform the duties of their jobs. In the sustainment phase, organizations have all the difficulty of hiring young people into government service on a hard-to-access government base that early programs have. Plus, new hires experience the added disincentive of not being able to help create a new thing; instead, they will patch and fix old things that they might never understand completely. This also serves as a disincentive to stay when one is experienced.

B8S1. Disparate commercial tools need to be cobbled together into a development/test environment.

B8S2. Program needs to rework facility environment to deal with higher communications bandwidth. Sustainment-phase organizations use the tools they have, which are often outdated. This is both because they are starved for a budget to invest in new tools (“Why should we spend money on old programs?”) and because the software that they maintain was built with such tools; using them may make more sense than upgrading them if doing so will require changes to the software. In addition, because sustainment lasts a long time, programs tend to outgrow their facilities.

B9S1. Government, contractors, and vendors argue over data rights. This is the same issue whether it occurs in the early lifecycle or during the sustainment phase. However, the transition to sustainment is the single most important time to have data rights spelled out clearly in advance.

B10S1. Contractor hands off to organic organization for sustainment. The amount of knowledge transfer is a function of the time that the two organizations spent working on the program together. In the sustainment phase, the contractor might be released from the contract and the work transferred completely to an organic responsibility. Issues such as continued presence of the contractor, data rights, and licenses need to be worked through.

B11S1. Contractor hands off to organic organization for sustainment. The amount of knowledge transfer is a function of the time spent working on the program together. This sustainment elaboration includes management issues associated with reversion of program responsibility to the government from the contractor(s), if this is in the program plan.

(NEW) B12. Information assurance/cybersecurity surprises require redesign or retrofit of HW/SW. A new change driver category applies to the security needs on programs during sustainment and maintenance. This relates slightly differently to sustainment than to early programs, in that the sole goal during sustainment is to fix problems rather than to design preemptive solutions. Since the sustainment phase is so long, it is virtually impossible to design a system that has enough security features to be good enough as is throughout the program.

Elaborations of C: Engineering Solution/Work Products Categories for the Sustainment Phase

C1S1. Information assurance/cybersecurity surprises require redesign or retrofit of HW/SW.

C1S2. Program experiences unfunded (or funded) capability creep; e.g., operational users change their minds. These two requirements issues require only slight sustainment-phase elaboration. For security, the early phase change driver discusses establishing a security plan and defining the security requirements. The sustainment phase includes dealing with surprises and knowing when a redesign is required.

C2S1. Information assurance/cybersecurity surprises require redesign or retrofit of HW/SW. This is mostly the same for early and sustainment phases, although the sustainment phase could have more emphasis on determining whether redesigning is necessary.

C5S1. System can no longer be sustained, so starts fresh (SW fragility). A sustainment phase elaboration includes determining when software must be redone completely.

(NEW) C10. Information assurance/cybersecurity surprises require redesign or retrofit of HW/SW. In the sustainment phase, a program might need to write a new change driver to deal with the technical aspects of security.

Conclusion

The purpose of this report has been to familiarize the reader with QUELCE, to list the current change drivers, and to introduce elaborations to the change drivers that apply to sustainment. The QUELCE method, described in the first section, uses information about a program, a list of typical change drivers, and the probability of their occurrence in a workshop to develop scenarios of change to the program. These steps are followed by analysis to calculate the chances of meeting any given cost target on the program. Change drivers are fundamental to this method: these are factors that might occur that would substantially change the cost outcome of the program. The second section introduces our current categorization of change drivers. The third section describes a set of elaborations to the change drivers for use with sustainment-phase programs. The QUELCE method results in a cost estimate represented as a probability distribution that a decision maker can use to understand the level of risk associated with the cost estimate. It also produces an executable model that a program can use to run alternative scenarios and update in the future for re-estimation purposes.

Acknowledgments

Bob Stoddard participated in the interviews of sustainment organizations. Robert Ferguson, Dennis Goldenson, Jim McCurley, Robert Stoddard, and David Zubrow were the main developers of QUELCE and reviewed this paper. Tamara Marshall-Keim provided excellent and timely technical editing and process guidance.

References

[Bliss 2012]

Bliss, Gary R. *Observations from AT&L/PARCA's Root Cause Analyses*. NPS Acquisition Symposium. May 2012. http://www.acquisitionresearch.net/files/FY2012/NPS-AM-12-C9P18Z04-145.pdf

[Bolton 2008]

Bolton, Joseph G. et al. *Sources of Weapon System Cost Growth: Analysis of 35 Major Defense Acquisition Programs*. RAND. 2008.

[GAO 2009]

Government Accountability Office. *Defense Acquisitions: Assessments of Selected Weapons Programs*. GAO-09-326SP. U.S. GAO. 2009.

[Hofbauer 2011]

Hofbauer, Joachim et al. *Cost and Time Overruns for Major Defense Acquisition Programs*. Center for Strategic and International Studies. 2011.

[PL 111-23]
Public Law 111-23. Weapon Systems Acquisition Reform Act of 2009. 111th Congress. May 2009.

[PL 111-383]
Public Law 111-383. Ike Skelton National Defense Authorization Act for Fiscal Year 2011. 111th Congress. January 2011.

Contact Us

Software Engineering Institute
4500 Fifth Avenue, Pittsburgh, PA 15213-2612

Phone: 412/268.5800 | 888.201.4479
Web: www.sei.cmu.edu | www.cert.org
Email: info@sei.cmu.edu

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

DM-0003544