Booz | Allen | Hamilton

Booz Allen Hamilton Inc.
Suite 900
121 South Tejon Street
Colorado Springs, CO 80903
Tel 1 719 387 2000
Fax 1 719 387 2020
www.boozallen.com

Automating SEER for Software: A Quick Cost Estimation Process

John Teal and Gary Hellenga

Abstract

Software estimates are useful tools for examining project scopes ranging from an entire system-of-systems program, down to a specific component of a small subsystem. In general, however, producing an estimate is neither quick nor easy. Even worse, maintaining the estimates as projects mature and undergo change is laborious – and consequently, tracking the progress of a project's estimates over the project lifetime is rarely effectively accomplished. What is desired is an estimation process that can respond with agility to changes in project direction and to the assumptions on which the estimate is based, without undue cost and effort in updating the estimates themselves. This paper will discuss a process for automating and speeding up the iterative development and maintenance of estimates using SEER, reasons for creating the process, and the benefits to the estimator.

Most estimators use SEER in interactive mode, performing the laborious and error-prone task of manually entering data into the SEER User Interface client. This is a time-consuming and sometimes frustrating segment of the software cost estimating process. Unfortunately most estimators do not realize that their desktop application employs a client-server architecture, and the server portion can be utilized with a variety of clients other than the SEER for Software user. Using an Excel-based process, the server mode can accept a sequence of commands from a text file or the Windows clipboard, and process them to create a SEER Project file, generate estimates, and produce output reports that can be exported to the same or alternate client application. If performed properly, this process only takes a few seconds. The goal of this process is to reduce demands of data entry, and to empower the estimator to focus on the role of a software estimating consultant and analyst.

Section 1: Introduction to parametric software cost estimating

Starting now, software cost does not have to be as difficult as it used to be! Recent developments mean that manually reentering hundreds of project data points – one of the most tedious aspects of parametric modeling – can be significantly reduced.

Cost estimation of software projects is certainly one of the more daunting niches of cost estimating because it requires obscure domain knowledge, familiarity with little known commercial models, and an appetite for details. To engineers, software development can be a complicated frustrating field and to cost estimators the sentiment is no different. But despite the drawbacks, software estimating is a field that cannot be ignored. Given the spectacular failure rates of defense software development projects, managers desperately need to understand the scope of their system and software estimates are useful tools for examining project scopes ranging from an entire system-of-systems program, down to a specific component of a small subsystem. Managers need to understand the impacts of schedule changes and requirements creep which highlights the importance of software cost models because, as always, the true common denominator across most projects is the cost.

There are several ways that commercial models are used to estimate costs of software projects. In a perfect world, the cost estimator has access to a set of software requirements documents that are completed and unlikely to change throughout the duration of the project. In this situation, the estimator is functionally competent to create lines of code estimates using these requirements documents and perhaps even cross-reference these totals with Function Point assessments. The estimator combines all of this information and uses his or her years of experience in order to create a realistic estimate. Reality check! Too often, especially in the defense market, the requirements documents are being written while the code is being developed, the estimator doesn't have functional understanding of software development, and the project manager has a keen interest in tweaking estimate variables in order to keep the costs on the low side of realistic. Information is always in a state of change and the estimator has to keep up -- instead of software development being a linear process, it becomes iterative. While there is nothing inherently wrong with an iterative process, the strains of constant updates can become a burden.

While many view software cost estimating models as a godsend to the cost community, these same people should also identify the tools' weaknesses. First and foremost, these models have traditionally required a tremendous amount of data entry. A typical program element in a software cost estimating model may have dozens of required inputs. Multiply that by three because many models also require a low, medium, and high input for each element. Then multiply this by the number of programs, components, and COTS units, and it quickly becomes apparent how much data there is to manage. Matters become more complicated when requirements, staffing situations, and other variables begin changing on a weekly basis. Few estimators can juggle multiple what-if scenarios and make these constant updates without an error or two! It is an unfortunate situation when a highly trained software estimator is reduced to the role of a data-entry chimp specialist, focusing on getting the thousands of details correct rather than adding value as a consultant and analyst to the project. Software cost does not have to be as difficult as it used to be! This paper will address, and attempt to solve, the problems commonly associated with the iterative style of software development.

Section 2: SEER for Software

Out of all of the software cost estimate models commercially available, this paper focuses on SEER for Software (until recently known as System Evaluation and Estimating of Resources – Software Estimating Model [SEER-SEM]), owned and licensed by the Galorath Corporation. In addition to software development costs, the model estimates maintenance cost, effort, schedule,

reliability and risk by comparing parameters that the user enters to a vast database of industry-wide historical data. SEER can generate a cost estimate even if complete project data is not available, because the model can leverage historical data and make assumptions based on what little data it is given. In the SEER cost model equations, estimates of new lines of code and pre-existing lines of code are critical for a cost estimate development, and knowledge of requirements volatility, the effort to re-host from the development environment to the target environment, security requirements levels and years of maintenance are all high priority items as well. However, there are almost 150 unique items in a Program element in SEER that require attention in order to perfectly calibrate the model to a specific project. A major development effort with multiple Program components may have thousands upon thousands of parameters to manipulate! In a recent NASA project the cost estimator was receiving frequent information updates and what-if drills from the software development team, and manually entering these thousands of parameters was not time-effective and occasionally resulted in data entry errors.

Resolving data entry errors is a time consuming process, and it ultimately has the effect of reducing the credibility of the estimator. Identification of data entry errors may occur when a current report is compared to a previous report and the cost is a few thousand dollars different, or the schedule duration is off by a couple of days. Finding the parameter that caused the delta can take hours if the estimate is big enough, although it can seem like an eternity when the project manager is looking on. Even seasoned SEER users find themselves in this unfortunate situation from time to time.

To speed up turn-around time and reduce errors, the team began using SEER's Server Mode. Within versions of SEER-SEM 6.0 and later, the server mode provides an alternative to manually entering data—WBS projects, elements, knowledge base selections, sizing data, and parameter values may all be created in SEER using server mode scripts, which are sequences of commands that can be stored in a text file. As Galorath describes in the SEER Help menu, the actual command strings that SEER requires are very similar to the different input parameters that SEER users are familiar with, except the strings consist of the parameter followed by a tab, followed by text or data; followed by any additional tab-delimited values. Where a command includes more than one value, the number of lines of new code for example, tabs delimit the values.

This is an example of a command script that creates 4 Program elements at the top level of hierarchy, with the last two having subordinate COTS and Component elements:

ProjectCreate [*tab*] MyProject
WBSCreate [*tab*] Parent 1 [*tab*] Program [*tab*] 1 [*tab*] [*hard return*]
WBSCreate [*tab*] Parent 2 [*tab*] Program [*tab*] 1 [*tab*] [*hard return*]
WBSCreate [*tab*] Parent 3 [*tab*] Program [*tab*] 1 [*tab*] [*hard return*]
WBSCreate [*tab*] Child 1 [*tab*] COTS [*tab*] 2 [*tab*] [*hard return*]
WBSCreate [*tab*] Parent 4 [*tab*] Program [*tab*]1 [*tab]* [*hard return*]
WBSCreate [*tab*] Child 2 [*tab*] COTS [*tab*] 2 [tab] [*hard return*]
WBSCreate [*tab*] Child 3 [*tab*] Component [*tab*] 2 [*tab*] [*hard return*]

Command scripts can be formulated from any appropriately-formatted text source, including text files and data already copied to the Windows clipboard. The default formatting applied to data copied from Excel worksheet ranges to the clipboard is perfectly suited for transferring commands from an Excel file to SEER.

The bottom line is that the estimator and the engineer can use Excel as a common way to keep track of input parameters, and when it's time to generate a new estimate it's as easy as copying the data range, opening SEER, and selecting Tools → Run Commands from Clipboard.

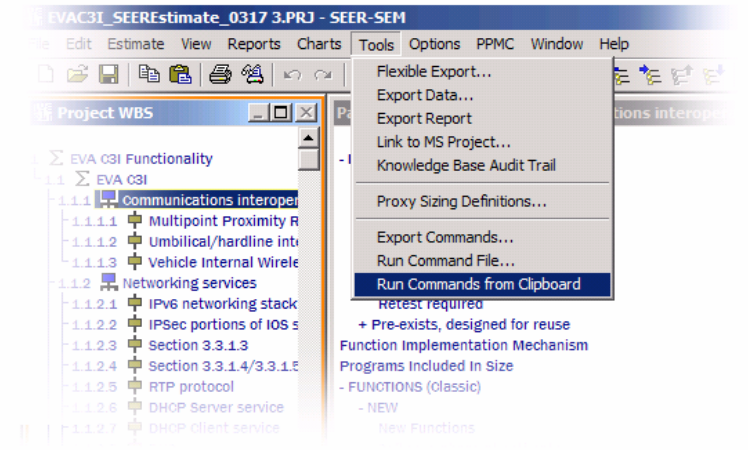

Figure 1: Run Commands from Clipboard capability

Although this capability represents a huge gain in efficiency, on a recent NASA project the team leveraged the premise of a client-server architecture and Excel's Visual Basic for Applications (VBA) programming language to further automate the process.

Section 3: Client-server architectures

Client-server architectures are prevalent in modern software design. The World Wide Web and other systems based on the same underlying protocols allow users with a Web browser client to interact with many different servers that can provide a variety of services, without the user's computer having to install specialized software for each of these services. In these situations, the domain-specific capabilities of the services reside on the server, and can be offered to anyone with a Web browser. This design offers considerable flexibility in distributing the services to a large number of users simultaneously.

The SEER tool also uses a client-server architecture. For most users, who run SEER as a desktop application where the client and server both run on the user's computer, this architecture is rather transparent – the application appears to be monolithic. But the advantages of using the client-server construct become apparent when SEER is integrated with other tools, such as Microsoft Project or the IBM Rational software engineering environment; with the same server application, SEER can support many types of client software in addition to its own native user interface. The nature of the client-server architecture is illustrated in Figure 2 below, showing how the SEER server accepts commands and returns estimate data to the client.

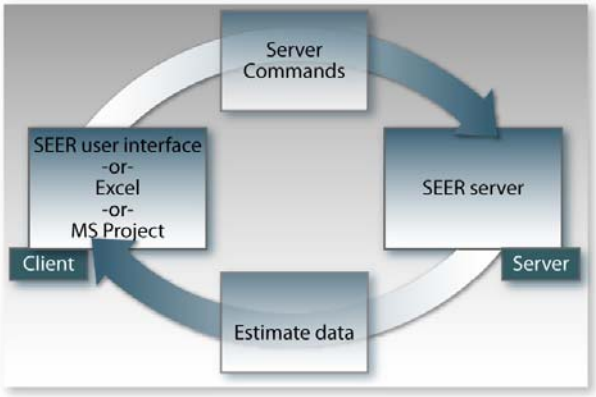

Figure 2: Interaction of the Client-Server system

One interface supported by the SEER server is a Microsoft automation (OLE) interface. Many Windows applications, such as Project, utilize this interface, and thus, integrating such applications with SEER can be done reasonably easily. In addition, the Windows clipboard can readily interface with SEER, and can be used to manually transfer data, commands, and results between various Windows applications and SEER. In the process Booz Allen has implemented, we use the automation capabilities of our customized Microsoft Office Excel spreadsheet to access SEER functionality, starting up the SEER server, if necessary, using it to create and populate SEER scenarios, then closing it when done. This allows SEER to be fully controlled from Excel, without having to manually run any SEER functions on the system.

With the ability to run SEER from Excel, the Excel workbook can serve as full-fledged client for SEER's server; almost all capabilities of the SEER desktop application can be duplicated by appropriate "coding" within Excel's VBA macro language.

Coding?! Within Excel?! Many users may think of Excel as simply a data-capture tool, but anyone who has used even the simplest formulas within a workbook, such as sums and other numerical calculations, should realize that some software processing of the data is going on behind the scenes. The formulas embedded in Excel utilize functions encoded in Visual Basic for Applications, a programming language used in most of Microsoft's Office suite of products, though users of products like Word and PowerPoint may have less use for this capability than Excel users. VBA provides built-in constructs that allow programmers to easily reference and manipulate common elements of Excel spreadsheets, such as Workbooks, Worksheets, Ranges (rectangular collections of cells on a worksheet), and the like. Combining this capability to access and manipulate sets of data with the ability to access and control SEER through an automation interface, makes Excel with VBA a powerful tool for capturing project data and building a cost estimate from it. Also, VBA provides the capability to create and use typical user interface "widgets", such as pushbuttons, radio buttons, checkboxes, and number spinners, to augment the worksheet cells as data entry and operations control facilities. These widgets are tied to underlying VBA subroutines, commonly known as "macros", that execute when the widgets are manipulated in certain ways (e.g., clicking a button, checking or un-checking a checkbox, etc.).

To an estimator who is not a proficient VBA programmer, the challenge to automate SEER may seem overwhelming; however an advanced knowledge set is not necessary. To help get users started, SEER includes an Excel file titled 'SEER-SEM Server Mode Details.xls' which is accessible by searching through the SEER directory that is created during the installation process. The workbook has information describing how to leverage the server mode capabilities of SEER. One section includes sample code to help the user learn how to use other tools such as Microsoft Excel or Project in concert with SEER. Specifically, the workbook includes samples of the VBA macro code needed to invoke SEER as well as the sample server mode commands needed to create an estimate. Furthermore, the service agreement for all of the major parametric estimating tools includes technical support to help users solve problems that might arise during the development process.

Section 4: Process Overview

The process described in this paper arose from work Booz Allen's Systematic Reuse Team performed in support of NASA. In several tasks performed under this engagement, the Team created cost estimates to help quantify cost differences between competing software acquisition strategies or competing system architectures. In the process of collecting the information about the systems involved, their characteristics, and their decomposition into subsystems and lower-level elements, the Team's software engineers and their NASA civil servant counterparts found it

natural to use Excel to capture this data; these spreadsheets served as collaboration tools for debating the salient aspects of these systems within work groups, as well as publishing conclusions to the outside community (Excel is used ubiquitously, and thus, data captured with it is generally easy for even outsiders to understand and review). Typical spreadsheet functions such as sums and averages were utilized to help understand each level of the resulting system hierarchies. As the Team began to use SEER to model the cost aspects of these systems, we found we needed to add a few additional calculations to derive the input values SEER required. Eventually, virtually all the numerical inputs needed by SEER were being calculated within the spreadsheet; other assumptions needed to produce estimates, such as project team capabilities, development environment and target system parameters, project schedule data, etc., were generally captured in Word or text documents, but could just as easily have been added into the Excel workbooks that held the numerical data.

Once the data of interest was captured, in one or more source documents, it was used to create SEER scenarios (Projects). Data from the Excel spreadsheets was used to create the project Work Breakdown Structure (WBS) and to supply size estimates for each WBS element. As the engineers debated spreadsheet values and made frequent changes to spreadsheet content, the SEER Project files had to be correspondingly updated. The process of manually copying spreadsheet data to the SEER user interface, and keeping the SEER models in synch with the spreadsheets, soon became tedious and occasionally error-prone. After the first errors were discovered, the Team adopted the paradigm of printing input reports along with all estimate reports, and manually comparing those input values with the spreadsheet contents – this "QA" function added even more time and tedium to the process of generating estimates. We realized what was really needed was an automated process that would streamline the transfer of the spreadsheet data to SEER, to reduce the time and eliminate errors associated with the manual process.

Just as this need was really becoming apparent to us, Galorath announced an upcoming webinar on integrating SEER with other tools. This webinar described the SEER client-server architecture discussed above, and while not focused on Excel as a client, it became apparent that the SEER Server Mode might offer our team exactly what we needed to address our data transfer issues. Working with Galorath's Tech Support team, we used Galorath-provided interface documentation and code samples to build the VBA macro code needed to automate our use of SEER from our Excel workbooks. Along the way, the team generalized spreadsheet data structures to allow them to be used across a variety of NASA systems and subsystems we were looking to support in our on-going work. The amount of VBA coding that was required was not overwhelming due to the technical support and examples provided by Galorath. Someone who has never programmed software would likely have a hard time re-creating this capability, but an analyst who routinely builds macros will probably not find this too difficult.

Now, anyone on our team can use the Excel template, with its specialized macros for interfacing with SEER, to capture the important aspects of a project in a single worksheet, and use user-friendly interface widgets to identify what is to be done with the workbook contents (such as generating an estimate with reports X, Y, and Z). This can then be passed to anyone with a licensed copy of SEER (the Booz Allen Systematic Reuse Team engineers do not have licenses to run SEER, but are paired up with cost estimators from the Economic and Business Analysis team who can advise them on the use of SEER and can make the actual runs of the tool); all the recipient needs to do to create an estimate is click the "Go!" pushbutton.

The process showing in Figure 3 below shows how the team can manage data in Excel and send the workbook to the estimator when updates are required. A macro alleviates the need for the

estimator to review the Excel file, open SEER and enter project updates—instead, a macro automates the new process and the Excel file is updated with new costs and reports in a matter of seconds. After a quick review the estimator can send this information back to the project team.

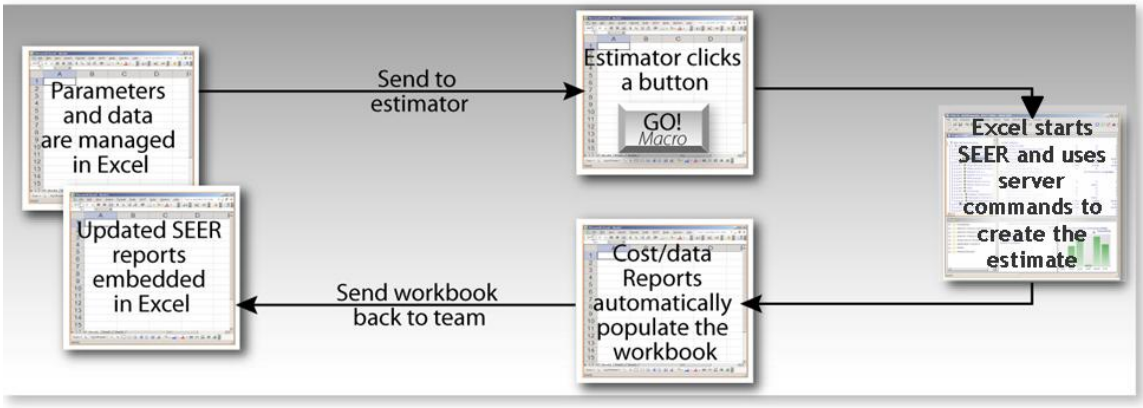

Figure 3: Automated Estimation Overview

The generation of the estimate, and export of estimate reports back to the Excel workbook, takes only seconds. This ability to create cost models in SEER so quickly and efficiently greatly improves the utility of the estimation process, as it affords the ability to do many 'what if' scenarios that might otherwise take too much time and effort, gives assurance that the entry of data into the models is free of errors, and provides such a convenient means of update (via the Excel workbook) that the estimate is more likely to be maintained and updated throughout the project's lifecycle. In addition, the cost analyst is freed to spend most of his/her time performing analysis on the inputs and results, instead of the mechanics of creating the model and entering data.

Section 5: Advanced Process Overview

While the ability to generate estimates in seconds with only a button click is tremendously powerful, there are situations where an even higher degree of automation is desirable. For example, what if the cost estimator is attending an all day software engineering meeting and is receiving quick 'what-if' taskings? One option would be for the estimator to run down the hall to his or her desktop every time a parameter changes and run back with new cost reports. Although that would work, there are ways to automate the round-trip processing of the spreadsheet file so that the software team can focus on capturing the pertinent attributes of the new estimate-- then generate the results in minutes-- using the Excel capabilities described above.

The Booz Allen Excel template allows setting of an "autoprocess" flag. When the workbook is opened, this flag's value is checked, and, if set, the "Go" functionality is automatically executed. On the cost analyst's computer, the team built and installed a scheduled process that runs every few hours. When the process is activated, it searches a particular file directory on a shared file server, and opens, then closes, any workbooks it finds there; after the workbook is processed, it is moved to another directory. The opening of the file causes the "Go" macro to be executed, and the estimate is created. All the software team has to do is access the shared file server during the meeting, submit an updated Excel file, then pick up the finished product after the automated process has run. The estimator never has to leave the meeting.

Section 6: Other Uses, Future Uses

Given how well the process worked here, where else can it be applied? First and foremost, SEER for Hardware (formerly known as SEER-H) is an obvious opportunity because it operates in a

near parallel fashion as SEER for Software. In fact, SEER for Hardware might be an even better candidate for the advanced automated process described in Section 5 because engineers often play a bigger role in identification of hardware component parameters than they do in the software realm, and would therefore want a more automated, iterative process. Pertmaster, a probabilistic schedule management tool, also can import the data required to define a plan such as the plan title, task names, task durations, resource assignments, start and finish dates. As with SEER, this data simply needs to be created in any sort of text file in a comma delimited text format. ACEIT, a popular cost estimating tool from Tecolote Inc., allows the user to import data from both Excel and Word using plug-in extensions. This process seems analogous but at this point the nature of the client-server architecture is unclear. While this paper may have read like an advertisement for SEER, realize that the beneficial capabilities of many common estimating tools may be 'hiding in plain sight.' Estimators can gain efficiency and improve accuracy if they are willing to find these capabilities and learn about them.

The benefits of using this approach are difficult to quantify for the NASA Systematic Reuse Team because the new process is orders of magnitude faster and more efficient than the manual mode the team employed previously. Whereas data entry used to be a full-time job for several weeks, plus several hours per week thereafter to include updates or perform what-if drills, many of these tasks now only take several seconds. The amount of efficiency gained though this process offsets the initial developmental effort, especially when considering that the development effort only occurs once but the efficiency can be actualized across many projects. Also, given that a mid-level cost estimator can frequently cost the government over one hundred dollars per hour, the total project savings is very substantial if the estimator requires less time to do his or her job. More importantly, with the reduced data-entry workload, the estimator has increased time to act as a consultant to the estimating team.

Biographies:

John Teal is an associate with Booz Allen Hamilton in Colorado Springs and works in the areas of cost analysis, life cycle cost estimating, business case analysis, cost risk analysis, and economic analysis. His six years of cost and finance experience have been on space and aircraft programs for the Air Force, intelligence agencies, DoD joint programs, and NASA. He is currently pursuing an MBA at Regis University and has a Bachelor of Science degree from the United States Military Academy at West Point. Mr. Teal is a Certified Cost Estimator/Analyst (CCE/A) and a Project Management Professional (PMP). He is a member of the Pikes Peak SCEA chapter where he serves as the Program Chair. He can be reached at teal_john@bah.com or 719-387-3757.

Gary Hellenga is an associate with Booz Allen Hamilton in Bozeman, Montana and is a software and systems engineer, performing both as a consultant and a practitioner. He has over 15 years of experience related to full software lifecycle aspects of space-related systems, including real-time flight software, flight planning and simulation, remote sensing modeling, image processing, communications system planning, and launch loads analysis. He has supported programs for the Department of Defense and several intelligence agencies, in addition to his current role supporting NASA. He has a Master of Science in Applied Mathematics from Montana State University, and is a Microsoft Certified Applications Developer. He has helped develop cost estimating processes within the Booz Allen Colorado Springs software development team, and has applied cost estimating techniques to business case analyses supporting NASA's developing Constellation Program. He can be reached at hellenga_gary@bah.com or 406-587-6177.