Improving Software Development Cost Estimation Models

Rodney Figaroa, Scott Nelson, Karen Williams
PEO STRI
Orlando, FL
Rodney Figaroa @us.army.mil
Scott.h.nelson1@us.army.mil
Karen.e.williams@us.army.mil

Charles Stroup
SAIC
Orlando, FL 32766
Charles.f.stroup.jr@saic.com

Arlene Minkiewicz, Bob Koury
Price Systems
Mt. Laurel, NJ
Arlene.minkiewicz@
Bob.Koury@
Pricesystems.com

ABSTRACT

Program Executive Office Simulation, Training and Instrumentation (PEO STRI) is utilizing a new code counting methodology to estimate future cost for software development products. In 2011, PEO STRI awarded two contracts in support of the One Semi-Automated Forces (OneSAF) program valued at more than $90M. For the first time ever, a requirement for the delivery of a Software Resources Report (SRR) was placed on each contract. The SRR is expected to be used to obtain the estimated characteristics of a software product and its development process. The intent of the SRR process is to collect objective measurable data commonly used by industry and Department of Defense (DoD) cost analysts. These data are used to compile a repository of estimated software product sizes, schedules, and effort that Government analysts can draw upon to build credible size, cost, and schedule estimates of future software-intensive systems. Information to be acquired through these data will include descriptive information about the product and developer and estimates of software product size, development schedule, peak staff, and direct labor hours. The paper will describe the Government's intent for use of the SRR, and describe the current state of this pilot program. The paper will detail the processes, the tool, participants, OneSAF unique challenges, methodologies and data. In conclusion, we will present the findings, lessons learned and recommendation for the future implementation of this product.

ABOUT THE AUTHORS

Rodney Figaroa is a Project Director at PEO STRI in Orlando, Fla. His primary responsibilities include overseeing the successful capability development for OneSAF. He has over 6 years of project management and engineering experience in the development, integration, testing, and fielding of simulation capabilities. He holds a B.S. in Computer Engineering from the University of Central Florida and a M.S. in Business Administration (MBA) from Webster University.

Bob Koury is the PRICE Systems Chief Solution Architect for Army accounts. In this role, he is responsible for ensuring that all Army customers have access to the cost estimating subject matter expertise necessary to be successful. Bob retired from the Army as a Lieutenant Colonel after 22 years of distinguished service and spent 18 years in industry creating Systems of Systems solutions for Texas Instruments, Raytheon and Lockheed. He holds a M.S. in Systems Management from the University of Southern California and a M.A. in National Security and Strategic Studies from the United States Naval War College.

Arlene F. Minkiewicz is the Chief Scientist at PRICE Systems, LLC with over 27 years of experience at PRICE building cost models. She leads the cost research activity for TruePlanning, the suite of cost estimating products that PRICE provides. She is widely published and speaks frequently on software related topics. She holds a B.S. in Electrical Engineering from Lehigh University and an M.S. in Computer Science from Drexel University

Scott H. Nelson is a cross functional Operations Research Analyst supporting several programs including OneSAF at PEO STRI in Orlando, Fla. His primary responsibilities include preparing cost estimates, analyzing Earned Value Management reports and managing Integrated Baseline Reviews. Scott has worked in private industry at Ford Motor. He is a Certified Public Accountant (CPA) and has over 20 years of expertise in Finance and Accounting functions. He holds a B.S. in Accounting and Master in Business Administration (MBA) from Brigham Young University.

Charles Stroup is the Lead Systems Engineer for the OneSAF Production Contract. He has been a systems engineer on the OneSAF program for the past ten years while working on the various SAIC contracts for OneSAF. He has more than 30 years of military simulation domain experience. He holds a B.S. in Aeronautical Science from Embry-Riddle Aeronautical University, a M.A. in Business from Central Michigan University, and a M.S. in Computer Science from West Chester University.

Karen Williams is the OneSAF Chief Engineer at PEO STRI in Orlando, Fla. She has more than twenty five years of engineering experience in the development, integration, testing, and fielding of simulation capabilities. She holds a B.S. in Physics from Jacksonville University and a M.S. in Industrial Engineering from the University of Central Florida.

Improving Software Development Cost Estimation Models

Rodney Figaroa, Scott Nelson, Karen Williams
PEO STRI
Orlando, FL
Rodney.Figaroa @us.army.mil
Scott.h.nelson1@us.army.mil
Karen.e.williams@us.army.mil

Charles Stroup
SAIC
Orlando, FL 32766
Charles.f.stroup.jr@saic.com

Arlene Minkiewicz, Bob Koury
Price Systems
Mt. Laurel, NJ
Arlene.minkiewicz@
Bob.koury@
Pricesystems.com

BACKGROUND

Working within the directives of the Deputy Assistant Secretary of the Army Cost and Economics Office (DASA-CE), Program Executive Office for Simulation, Training and Instrumentation (PEO STRI) typically applies a growth factor to software cost estimates when local, calibrated historical databases are not available to be used as the basis for estimating a future software development effort. As a result, software development estimates can be overstated by as much as 30-60%, negatively impacting the ability to obtain approval for execution of the project in a timely and affordable manner. In preparation of these estimates, a significant amount of time is expended by PEO STRI team members to develop and justify software cost estimates. While processes for capturing software development costs exist, the lack of a standardized process and system for collection of this data leads to a large reliance on Subject Matter Experts (SMEs) to provide subjective estimates of future development costs.

Acquisition Category (ACAT) I programs (valued at greater than $365 million RDT&E) require a periodic Software Resources Data Report (SRDR) to be delivered by the prime contractor. The SRDR contains various details on actual Software Lines of Code (SLOC), programming languages, commercial or government off-the-shelf (COTS/GOTS) applications, external interfaces, requirements, peak staff, and direct labor hours. This data is used to compile a repository of software product sizes, schedules and effort that Government analysts can draw upon to build credible size, cost, and schedule estimates of future software-intensive systems. The intent is for PEO STRI to create a document similar to the SRDR that would accommodate reporting by smaller programs (i.e. ACAT II). PEO STRI selected the ACAT II One Semi-Automated Forces (OneSAF) program to serve as the pilot for implementing this new reporting requirement.

OneSAF

OneSAF is a composable, next generation Computer Generated Forces simulation that represents a full range of operations, systems, and control processes from an individual combatant and platform to brigade level. It includes a variable level of fidelity that supports all Modeling and Simulation domains by accurately and effectively representing the specific activities of ground, air, sea, and space warfare (engagement and maneuver); Command, Control, Communications, Computers, and Intelligence activities; and combat support/combat service support missions as described in Figure 1. It employs appropriate representations of the physical environment and its effect on simulated activities and behaviors.

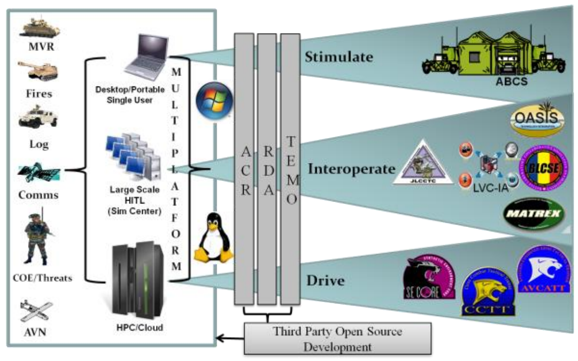


Figure 1. OneSAF Mission

The OneSAF program is currently in the production and support phase of the life cycle with ongoing implementation of Pre-Planned Product Improvements (P3I) as prioritized by the US Army Training and Doctrine Command OneSAF Project Office. Through approved P3I efforts, capability enhancements are continuously being developed and integrated into the software baseline resulting in, as a minimum, yearly version releases. OneSAF is fielded to multiple Army users as well as other Department of Defense agencies, industry, academia, and Foreign Military Sales cases. Concurrent with P3I enhancements, customer requirements, user feedback and needs are continuously addressed by way of Change Requests and Engineering Change Proposals to assure maximum utilization of the system throughout the growing user community.

OneSAF, as a software only program, was selected as the prime candidate for implementing, demonstrating, improving, and expanding the utility of code counting tools used to generate a standardized SRDR like report called a Software Resources Report (SRR). OneSAF consists of more than 6 million traditional SLOC and is continuously being revised, expanded and improved as capability and user community needs are addressed. Version 1 of the software was officially released on Sept 29, 2006, with more than 50 subsequent major and minor releases addressing domestic, domestic minus and international communities and requirements. Version 6.0 is scheduled for December 2012. Two new OneSAF contracts were awarded in 2011; one for OneSAF Integration, Interoperability and Support (I2S) to Cole Engineering Services Inc (CESI). and the other for software Production to Science Applications International Corporation (SAIC). With the timely renewal of these contracts, a requirement for delivery of a SRR was placed onto these contracts.

PROCESS

Figure 2 provides an overview of the process that evolved. The top layer of Secretary and Department of the Army level organizations represent the ultimate consumers of the data. It is the intent of PEO STRI to use the data collected as part of a budget approach which uses historical data to calculate defendable cost estimates. The middle layer of organizations is internal to PEO STRI and represents the data users. The Chief for Financial Management will initially use the data as direct inputs into the cost estimating software to calculate budget submission. In the future, the data may be adapted to create new cost estimating relationships or as a means to validate EVM metrics reported.

The lower section of Figure 2 provides an overview of the process flow specific to this paper. The OneSAF Program Office, OneSAF developers, PEO STRI Cost Division and PRICE Systems L.L.C worked as an Integrated Product Team (IPT) to develop the process and mature the tools for executing the collection and application of the software development resource data. The Program Office provided access to the software development contractors and overall guidance regarding the collection of the OneSAF data. PEO STRI Cost Division guided the process development strategy. PRICE Systems L.L.C. contributed cost estimating subject matter expertise and recommended the tools for the developers to use to produce the data. The team ultimately created a repeatable data collection process by modifying an open source SLOC counting application, creating a Data Dictionary and refining the original SRR form to be used in the collection of data.

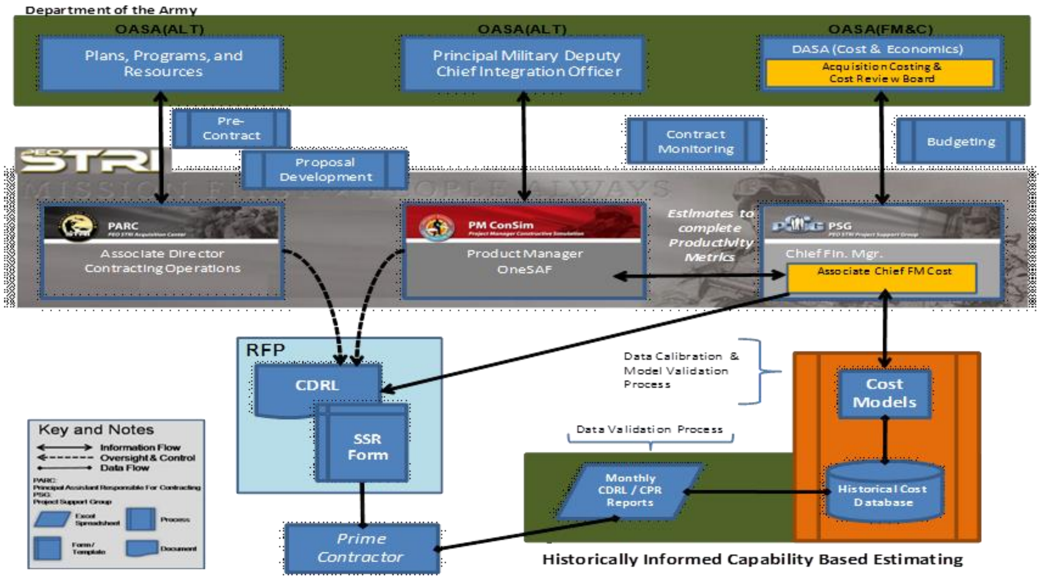


Figure 2. Defendable Cost Estimate Process

Frequency of Reporting

One of the first things the IPT addressed was the frequency of data collection and reporting. The Government originally placed a monthly reporting requirement on the contracts. Continued discussions about collection and reporting highlighted that a monthly report cycle had no real value and just increased costs to deliver a Contract Data Requirements List (CDRL). OneSAF conducts software development in blocks of time that are called ‘builds’. A typical build consists of 10-12 weeks of software development activity from requirements analysis through integration and test. During any build cycle, there are any numbers of new capability development, system integration, test and release activities occurring in parallel. The integration of capabilities back into the main OneSAF baseline typically does not occur during the same build because they all vary in size and complexity. For that reason, the IPT determined that the SRR report be build based and that the reporting scheme be software capability based. This reporting pattern offered traceability of SLOC and labor hours across builds and through the implementation, integration, test and release lifecycle for each capability. Figure 3 offers a graphical depiction of the what, when and how this nuance is captured in the report. At the end of each build, the contractors will collect SLOC and labor hours data per newly developed capability and integration activities for each software capability. The data reported for the integration activity will only include associated labor hours for performing this task and any SLOC changes to enable the integration back into the main baseline. The integration effort will not report the SLOC associated with the development phase. This allows for segregation of the code effort in a way that can determine productivity levels during all phases. Cost collection on any software capability is not considered complete until it is officially released. When the capability set for the next release is defined and the software baseline frozen, the program executes a version release cycle. The labor hours and SLOC changes necessary during this release cycle to enable a version release is captured and reported per capability into the SRR report. This last portion of the report is referred to as the final software build. Upon receipt of the final Software Build for each annual release, a Release Report will be developed that captures cost and capability per OneSAF release. With this approach, the database would ultimately contain all of the information necessary to make estimates based on capability.

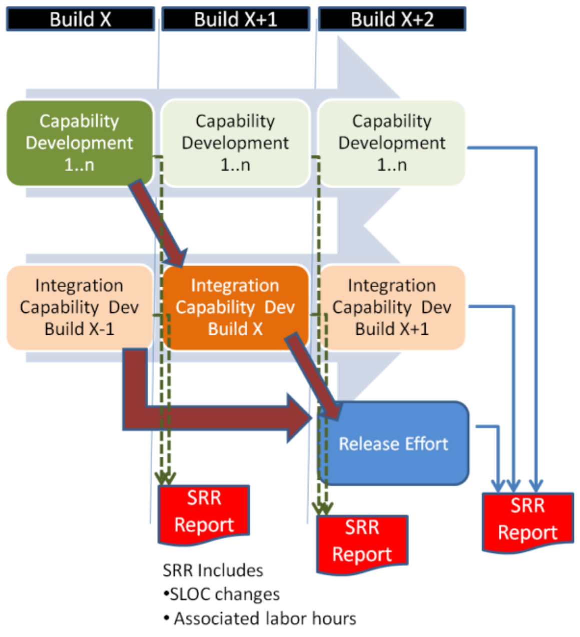

Build X
Build X+1
Build X+2
Capability Development 1..n
Capability Development 1..n
Capability Development 1..n
Integration Capability Dev Build X-1
Integration Capability Dev Build X
Integration Capability Dev Build X+1
Release Effort
SRR Report
SRR Report
SRR Report
SRR Includes
•SLOC changes
• Associated labor hours

Figure 3. Report Structure to Capture Effort by Capability

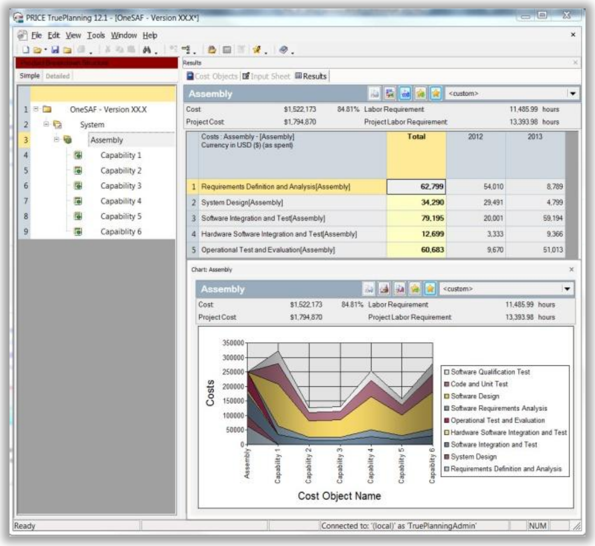
Software Cost Estimating and Reporting Tools

The TruePlanning Software Tool was selected to be the primary cost estimating tool. The intent was to ensure that the data collected would be easily used as input in the tool but also support the use of other commercial tools. The selected tool is a cost estimation framework that houses cost estimation models for hardware, software and systems as pictured in Figure 4. This framework encompasses most of the capabilities that are necessary or useful to cost estimators in the preparation, analysis and presentation of their cost estimates while the estimation models contain the Cost Estimating Relationships (CERs) and logic to perform estimates of effort, cost and schedule for a particular project. The framework provides the user interface for the cost models, creating input sheets and worksheet sets based on the activities, resources and inputs defined in the cost model. The framework contains the mechanism for output of the cost model results through tables and charts in a variety of user specified formats. It also contains a set of utilities providing capabilities common to many cost models such as the application of escalation rates and other economic factors, the application of cost schedule impact when schedules are constrained or stretched, and a cost risk analysis tool. The tool has evolved over time and improved to estimate the design, code and test implications of object orientation, model new software technologies and estimate using metrics in addition to source lines of code. Additionally, it includes new ways to measure reuse and properly estimate all of the costs associated with building software that uses COTS components.

The tool's software model is activity based. This means that the model uses the language of cost objects (products or services being developed or delivered), activities, resources and cost drivers. The cost objects are Software Components and Software COTS. Software Components represent any piece of software that is being developed in-house. Software COTS represent any piece of software that comes from an outside source (bought or furnished) that is to be integrated with other pieces of a system. The organization is such that the costs and effort estimated for each cost object are broken down and presented by activity and resource categories – so the user gains an understanding of how each activity and resource fits into the context of the entire project.

Each cost object has a group of cost drivers for which the user must enter values. These values are then applied to relationships that indicate how variances in this input impact the productivity on each of the cost objects activities. The Tool assigns a baseline productivity factor for each activity based on industry standard data. Productivity adjusters are applied to the baseline and then the adjusted productivity is applied to each activity for a particular project. The same process is used to determine how the activity requirement is spread throughout the resources.

For the software estimation model, the major cost drivers include software size, functional complexity, operating platform, and characteristics of the development team and organization. Software size is typically measured in SLOC and is strongly correlated to cost. Functional complexity is associated with the types of functionality the software is intended to deliver based on the assumption that some functionality is easier to design, code and test than other functionality. For example, real time command and control software is more complex than software that performs simple data manipulation. Clearly, the experience and knowledge of the development team will be a significant cost driver, but it's also important to note that organizational culture and practices will likewise impact software development costs. Another factor with significant impact on cost is the platform or environment in which the software will operate. For example, avionics software requires more rigor throughout the development process than software performing office automation.


PRICE TruePlanning 12.1 - [OneSAF - Version XX.X*]
File Edit View Tools Window Help
Results
Simple Detailed
Cost Objects Input Sheet Results
1 OneSAF - Version XX.X
2 System
3 Assembly
4 Capability 1
5 Capability 2
6 Capability 3
7 Capability 4
8 Capability 5
9 Capaiblity 6
Chart: Assembly
Costs
Cost Object Name
Assembly
Capability 1
Capability 2
Capability 3
Capability 4
Capability 5
Capaiblity 6
Software Qualification Test
Code and Unit Test
Software Design
Software Requirements Analysis
Operational Test and Evaluation
Hardware Software Integration and Test
Software Integration and Test
System Design
Requirements Definition and Analysis
Ready
Connected to: '(local)' as 'TruePlanningAdmin'
NUM

Assembly <custom>

Cost	$1,522,173	84.81%	Labor Requirement	11,485.99	hours
Project Cost	$1,794,870		Project Labor Requirement	13,393.98	hours

	Costs : Assembly - [Assembly] Currency in USD ($) (as spent)	Total	2012	2013
1	Requirements Definition and Analysis[Assembly]	62,799	54,010	8,789
2	System Design[Assembly]	34,290	29,491	4,799
3	Software Integration and Test[Assembly]	79,195	20,001	59,194
4	Hardware Software Integration and Test[Assembly]	12,699	3,333	9,366
5	Operational Test and Evaluation[Assembly]	60,683	9,670	51,013

Figure 4. Cost Estimating Tool

Collection Tool

It became apparent that there was a need for a standardized collection tool. The standardized tool would allow for consistency in code counts from one software development Build to the next and from one program to the next. A very important component to the success of this pilot effort is that the contractors need the ability to perform consistent code counts in a reasonable amount of time. Thus, an important success factor was the automation of the process as much as possible. The original plan was to use the freely available code counting tool developed by the University of Southern California Center for Software Excellence (USC-CSE) called the Unified Code Count (UCC). This tool implements popular code counting standards published by the Software Engineering Institute (SEI) and adapted by the Constructive Cost Model (COCOMO). This tool counts both logical and physical lines of code and performs code counts of 21 software languages. In addition to counting code, the tool has source differencing capabilities. This makes it possible to count, compare and collect logical differentials between two version of the source code of a software product. This fact was key to automating the ability to identify new, deleted, changed and unchanged code from build to build.

The code counting tool counts both physical and logical lines of code. A physical line of code is just what it sounds like – one single line of code on the page or screen. A logical line of code represents one single command to the compiler. Many programming languages are flexible enough in their syntax that it is unwise to assume that each physical line of code represents a single command to the compiler. Additionally, the cost estimating relationships in the software model are based on an analysis of projects with logical line of code counts. For these reasons, the decision was made to count logical rather than physical lines.

The deployment of the UCC tool was not completely smooth. Because one end goal is the ability to determine effort at a capability level, it was important to find a way to align code counts with the specific capabilities being delivered with each build. An Excel based Code Count Wrapper (CCW) was created for the tool that allows users to perform differential analyses between two code bases. The CCW allows users to configure the types of files being examined, the directories of the two code bases, the depth to start within those directories, as well as configuration parameters that allow users to control the size of the runs of the tool and the ability to exclude specific directories from examination. The CCW also addressed the concern for the code count tool to process the large OneSAF codebase. The solution was to have the CCW identify all directories that contain files of the types selected, and then run the counter tool multiple times feeding off of the selected directories with each run of the tool being based on a CCW pre-configured total file size per run. Once the runs have been completed, the CCW gathers the results from the multiple runs and produces a report in Excel with the result broken out by language.

Characterization of the Code

OneSAF 'code' consists of a large amount of Excel and CSV type files. Originally it was believed that it was unnecessary to count these files as they were not technically traditional code and appeared to be merely data entry elements. However, it became evident that these files were indeed an important part of the work products that were being delivered which required significant investment of time and effort on the part of the contractors. For this reason, they became an important part of the cost/effort history of the project to capture.

The collection tool currently does not have the capability of performing line-by-line analysis of the modification /work performed on .csv and .xls files. Furthermore, there is no means to align these counts with skill level or labor hours; therefore the focus was placed on the magnitude and direction of the overall size of each file during the scanning. To offset any margin of error injected by this methodology, both .xls and .csv files are to be treated as "non-executable code" and weighted differently than executable code files. The team developed functionality in the CCW to count the change in number of rows (xls) and/or lines (csv). Each type of data file (xls, csv) is recorded as a summary number which represents the aggregation of both added and deleted data records. This is recorded as a separate capability in the product size reporting section of the SRR.

The tool calculates an amount of effort consumed by each resource for each activity. The amount of consumption of effort is based on a study of productivity for a predetermined work package size. The consumption of labor hours for each activity for a standard work package has been validated in the past using historical data. However, there is a continuing need to revalidate as code methods, tools, and processes change/improve. One key to the success of this data collection effort is to collect, in a methodical manner, the data necessary to link labor hours to activities to SLOC. The number of work packages is determined by dividing the total code count by the standard work package size:

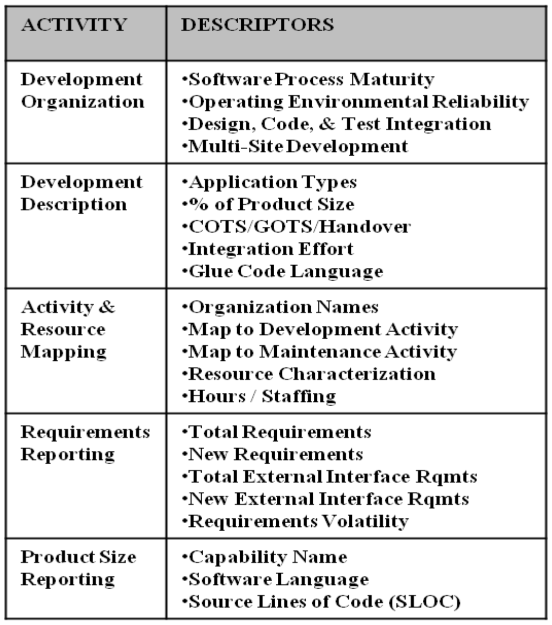
Number of Work Packages = Total Effort Size (Count)/Work Package Size

Non executable code is added to the total effort size at the rate of 33%. For example, if the non-executable code is 1000 lines, the tool adds 333 lines to the total effort size automatically as part of the count. Therefore the number reported by the tool in the summary line will already have been factored accordingly, or reduced by 67%.

Software Resources Report

The detailed reporting requirements for the SRR were refined as an output of the IPT. The final revised SRR was enhanced to address specific OneSAF nuances as related to Development Organization, Development Description, Activity & Resource Mapping, Requirements Reporting and Product Size Reporting. Although these nuances were OneSAF specific, the team determined that they were a worst case situation which facilitated the creation of a set of tools, and processes which would meet most collection requirements. Table 1 highlights what and where this new information is to be addressed within the report. The SRR form was initially designed to meet the vision of how an agile software development program should report its development status in terms of SLOC and labor hours. After several working sessions and numerous emails, the form as it exists today provides a template for reporting on the breadth of software development program types from waterfall to agile, and does so in a way that provides meaningful data.

Table 1. SRR Data



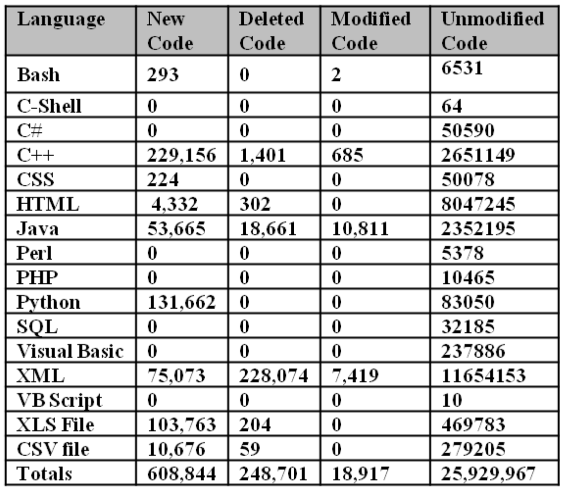
ACTIVITY	DESCRIPTORS
Development Organization	•Software Process Maturity •Operating Environmental Reliability •Design, Code, & Test Integration •Multi-Site Development
Development Description	•Application Types •% of Product Size •COTS/GOTS/Handover •Integration Effort •Glue Code Language
Activity & Resource Mapping	•Organization Names •Map to Development Activity •Map to Maintenance Activity •Resource Characterization •Hours / Staffing
Requirements Reporting	•Total Requirements •New Requirements •Total External Interface Rqmts •New External Interface Rqmts •Requirements Volatility
Product Size Reporting	•Capability Name •Software Language •Source Lines of Code (SLOC)

Findings

As of the date of this paper, the initial set of data has been formally delivered for OneSAF Build 28 activities and is currently being analyzed to verify the validity of the data and determine additional changes that may be required for the collection strategy.
The Build report covered activities over the ten week period from 5 March 2012 through 11 May 2012. The type of software development and maintenance activities for the Build included: development of approximately fifteen new from-scratch capabilities; integration of capabilities from internal co-developer teams; integration of two external co-developer handovers (one small and one large); and the correction of approximately 200 software issues.

Table 2, Initial Data, provides the results of running the code count tool on the OneSAF baseline comparing the start of Build 28 to the end of Build 28. The code count was performed on the 16 languages listed.

Table 2. Initial Data



Language	New Code	Deleted Code	Modified Code	Unmodified Code
Bash	293	0	2	6531
C-Shell	0	0	0	64
C#	0	0	0	50590
C++	229,156	1,401	685	2651149
CSS	224	0	0	50078
HTML	4,332	302	0	8047245
Java	53,665	18,661	10,811	2352195
Perl	0	0	0	5378
PHP	0	0	0	10465
Python	131,662	0	0	83050
SQL	0	0	0	32185
Visual Basic	0	0	0	237886
XML	75,073	228,074	7,419	11654153
VB Script	0	0	0	10
XLS File	103,763	204	0	469783
CSV file	10,676	59	0	279205
Totals	608,844	248,701	18,917	25,929,967

The results obtained from the code count tool must be coupled with the additional information provided in the SRR report in order to understand and provide context to the data. For example, the 25M count above includes non-executable code which is typically not counted. Additionally, the increase in Python and C++ (actually C) code is the result of adding COTS software for use in a new OneSAF capability and is not developmental code. The increase in the XLS file count and the decrease in XML file count is due in part from the conversion of OneSAF data between those file formats. This context needs to be captured in the database for future analysis.

The team plans to adjust the data collection strategy as needed in order to obtain useful data for future use. For this initial set of data the code count was performed on the complete code base exported from the Subversion configuration management tool. As the team is able to analyze the data more, it may be decided that portions of the code base should be excluded from the code count in order to provide more meaningful results. The CCW tool supports this by providing the capability to exclude directories from examination.

The CCW tool itself was found to be relatively easy to use once it was configured and the source code to be counted was obtained. The team worked through a number of issues in order to be in position to run the code count and submit the SRR data for Build 28, some of which are described in the Lessons Learned section. It should also be noted that the use of CCW and UCC allowed for broader language coverage than done previously on the OneSAF program, which focused on the primary developmental languages of Java and C++.

LESSONS LEARNED

The IPT worked together for six months; defining, refining and conducting test runs on sample data to achieve this first delivery of data. The following lessons learned are offered as insight as the process matures to meet the intended requirement.

Software Cost Estimating and Reporting Tools

The cost estimating and reporting tools went through multiple iterations. Versions that were created and working in the test laboratory were then tripped up by configuration issues in the contractor's facility. Initially the tools took a long time to run, six to eight hours, depending on the number of new, modified or changed lines of code. Performance fixes have since reduced the run time to about two and a half hours. In spite of this performance improvement, the code collection tool monopolizes Excel for the complete two and half hours. No other work can be done on Excel while the program is executing.

Language Types

In addition to the enhancement of the UCC tool to address .xls and .csv type files, the tool still does not address how to count terrain database code in any way. To date, OneSAF contains more than forty terrain databases and associated visualization files that are not being included in the cost collection process. Great effort is expended to generate, modify and debug these terrain databases. Generation of these databases requires the use of Digital Terrain Elevation Data (DTED), Vector Map (VMAP) and Ultra High Resolution Building (UHRB) specifications. Both DTED and VMAP come in a variety of levels of details, and the use of higher fidelity data often correlates with more effort being expended to generate and verify OneSAF models can reason off the database feature data. At the macro level, the team is still working on finding a relationship between DTED and VMAP level used, complexity of urban areas (in terms of UHRBs), number of feature data (roads, rivers, etc) and the size of the database (in terms of geotiles) to be able to estimate resourcing requirements for the generation of a database and develop an equivalence algorithm to be able to compare terrain database development to code development.

Subversion Issues

The OneSAF program uses Subversion, abbreviated SVN, as its configuration management tool. Most of the time, SVN executes all of its commands flawlessly. However, issues with SVN have surfaced when executing the Switch Command. The Switch command allows the user to "update" a checkout from one branch to another and is not changing to a new revision number entirely as expected. The Switch command ignores a directory that should have been updated. Because of this, a more brute force method of checking out both the beginning and end points of a development branch has been implemented, which can take as long as an hour and a half per checkout. OneSAF typically has 10 – 14 development branches for each development cycle adding as much as 35 hours to the SLOC counting effort.

Characterization of Data in the SRR

The SRR narrative that provides context and explanation of the code count results is important to the overall understanding of the data and necessary for future use. The initial SRR has been provided but it is unclear if the narrative will meet the needs of the intended users of the data. The team should continue to refine the information provided as the intended uses for the data is further understood.

Mapping Development Activities to SRR Standardized Software Activity

The SRR data is collected on a set of standard software activities. The program development activities, as defined in the Contractor Work Breakdown Structure (CWBS), need to be mapped to that standardized set. The mapping was for the most part straightforward and aligns well with industry standard software development. However, there are a few activities outside of main software development, such as overall system/software architecture, system analysis and design, information assurance, and development environment support that do not map directly to the SRR standardized software activities. The effort for these activities is being mapped in the Project Management and Control activity along with program management office effort. Future revisions of the SRR Data Item Description (DID) should add additional activities or include more guidance on suggested mapping to ensure all programs provide consistent data.

FUTURE WORK

Data is just data until we process it and turn it into useful information. The specific goals of the team are to create a situation where actual products can be referenced as the basis of project estimates and to be able to understand cost on a per capability basis so that informed tradeoffs can be made when there is insufficient funding for all the requirements. The SRR is designed to collect the data necessary to continuously validate the work package to effort relationship. This linkage will greatly improve the quality, accuracy, and defensibility of future cost estimates. The SRR facilitates the collection of characteristics of a software product and its development process. The team proposes the following next steps be taken to achieve the full vision.

Creation of a Benchmark Database

Although the DID and SRR were designed with a specific cost estimating tool as the target user of the data collected, the team always included the requirement to provide data to multiple cost and program management tools for a variety of uses. The data collected and housed in the relational database will be used to create a benchmark database for cost estimation. The definition of the data and the manner in which the SRR was structured anticipated the creation of a relational database as the warehouse for the primary software data. Specifically the team tried to design the collection process to allow multiple tools to link labor hours to effort (SLOC) to capability. Other data elements such as team maturity as measured by CMMI level, personnel experience, development site characteristics, capability name/category, etc. are collected in order to allow classification and retrieval of data from the database in forms required by multiple tools and users. This database could then be used to develop a benchmark database extract for the tool input values for ACEIT, or the creation of a new cost estimating relationship. The Software Effort Database is the focal point for the application of the data collected.

One could create data objects in the actual tool framework. Data objects are similar to cost objects except they contain no CERs. All of the data elements collected are inputs and the effort values are then translated to the appropriate outputs. As a simple example, if the SRR was collecting only New Size, Deleted Size, Design Effort and Programming Effort, then the data object would have as inputs New Size, Deleted Size, Design Effort and Programming Effort. The activities of the data object would be Design and Programming and the values input for Design Effort and Programming Effort would be throughput to the outputs on a ‘run’ of the data object. The data object could apply to entire projects, individual capabilities or both. The benefit of storing the data in a framework based database is that all the utilities and features designed for analysis and presentation would be instantly available.

Calibration of the Tool

The data should be used to calibrate the tool itself. The tool has been developed using data from many industries and many projects. This makes it a very general purpose model. The many possible input parameters contribute significantly to tailoring the model to a specific organization’s or projects particulars but calibration takes it a step further. The tool can be thought of as a meta model or a model for a model. Consider a model of the general form “a*X^b, if the calibration affects both the coefficient and the exponent, then the model becomes completely tailorable to an organization through the calibration of the input parameter that affects both coefficient and exponent. The Organizational Productivity input is intended to model the efficiency with which an organization delivers software. The process of calibration basically requires running the model in ‘reverse’ with the effort as an input and the Organizational Productivity as output. The calibration process is automated through the Excel Solution. As each release is calibrated over time, the results should converge to a steady state value. This value can then be used to estimate future projects.

CERs

The data could be used to create custom CERs implemented in the framework. Once several releases of data have been collected, analysis can be performed. Because the Data Dictionary and the SRR are being used by all parties to collect data, the data should already be normalized (apples to apples format across releases and contractors). Trend lines can be used to identify cost drivers and regression analysis can be performed to determine likely cost estimating relationships. This analysis could take place at the project or the capability level – a determination to be made after several iterations of data have been collected and analyzed. Once CERs are developed, they can be implemented as custom cost objects within the tool. As future iterations of data are collected, they can be used to validate that the CERs are still appropriate or to refine the cost estimating relationships to reflect changing development conditions.

CONCLUSION

The intent of the SRR process is to collect objective measurable data commonly used by industry and Department of Defense (DoD) cost analysts. These data are used to compile a repository of estimated software product sizes, schedules, and effort by capability. This repository will enable Government analysts to build credible size, cost, and schedule estimates of future software-intensive systems. Much time and effort has gone into creating a process and set of tools for data collection in order to ensure that data collection is relatively easy to do and not too time consuming. The team has also invested time and effort to create processes and tools that are not unique to the OneSAF program, or even PEO STRI. The goal is that they should be generally applicable to programs across the Army and the DoD with relatively little need for tailoring and customization.

While the pilot is still in its infancy, a great deal has been accomplished. Data collection has been added as a requirement of the contract, highlighting the Government's understanding of the importance of historical data to future projects. An agreement has been reached between all parties as to what data should be collected, how it is to be collected, and how often that data collection should occur. Tools and processes have been developed, tested, reworked, and put into place to facilitate the first formal data collection on Build 28.

The entire team has collaborated closely in an environment of cooperation and dedication to the success of the project to create a tool that automates a process for consistent collection of software size measures. The tool has evolved throughout the project. Requirements have evolved relating to what did and didn't need to be counted. No doubt issues will continue to arise as new, previously unconsidered, situations arise. PEO STRI is continuing with this pilot project and all parties remain committed.

ACKNOWLEDGEMENTS

The authors would like thank Mr. Vidal Acevedo for his enduring patience and insight throughout this process and Mr. Howard (Sam) Felder for his valued input on this paper.

REFERENCES

Army Regulation 5-11, Management of Army Models and Simulations

Department of the Army Pamphlet 5-11

California, U. o. (n.d.). USC Center for Systems and Software Engineering. Retrieved June 14, 2012, from Center for Systems and Software Engineering Code Count: http://sunset.usc.edu/research/CODECOUNT/