

Software Estimation

Developing an Accurate, Reliable Method

Bob Sinclair, NAWCWD

Chris Rickets, NAWCWD

Brad Hodgins, NAWCWD

Abstract: From a management perspective, it is essential that software estimates used in a TSP launch are as accurate as possible. Significant growth due to estimation inaccuracy can wreak havoc on a team attempting to stay within cost and schedule while executing its established plan. This article discusses how a software team that uses both proxy-based and size-based estimates is able to accurately plan, launch, and execute on schedule.

Introduction

The AV-8B Software Development Task Team has successfully maintained and enhanced avionics and support products for the Harrier II aircraft for the better part of a decade. While there are several factors that contribute to its success, a key element is the team's ability to provide timely and accurate cost and schedule estimates to its management and customer. This was not always the case. When the team first began preparing software estimates, it was ad-hoc. At that point, neither the Software Development Task Team nor its management had faith in the estimates. When the team adopted the Team Software ProcessSM (TSP)/Personal Software ProcessSM (PSP), it became a priority to define and document accurate estimates. In order for a team to execute a successful TSP/PSP project, the tasking estimates need to be well defined and communicated. If not done, the team will not buy into the resulting schedule and plan which could put the project in jeopardy of failure.

Background

The Naval Air Warfare Center Weapons Division (NAWCWD) AV-8B Joint System Support Activity has successfully applied TSP/PSP for software development and maintenance projects for nine years.

This began in the spring of 2002 when the software development task team began the H2.0 block upgrade maintenance software effort [1]. Since then the software team has completed an additional four block development efforts (H4.0, H5.0, H5.1, and H6.0) and is currently working the H6.1 block development effort. The block efforts typically last approximately two years and incorporate the TSP/PSP framework.

Up until 2002, all estimates were performed by a single individual, the lead software engineer. These estimates did not follow a documented process, much less a proven method. The estimates were rough and relied on engineering judgment (i.e., the estimates were prepared using the old "thumb to the wind" method). It was up to the team to develop a consistent estimation process. However, several questions needed to be addressed as part of this effort: How would the team determine the accuracy of its estimation approach? How would they know if the estimate was complete? Would something be missed? Could a reliable schedule that the team could execute against be produced from a set of detailed estimates? To compound matters, the team found out that the program office required multiple types of estimates. These estimates were needed to support the team's management in making budgeting, planning, and build decisions.

Types of Estimates

TSP projects are initiated by a project launch. This is a four- or five-day workshop where the TSP project team develops the project plan. Key roles, goals, objectives, requirements, and constraints are established during this workshop. Most importantly, for this discussion, the team establishes a detailed estimate and an overall project schedule [2]. Therefore, the software team's launch success was predicated on the team's ability to have an accurate reliable method of performing estimations for which to generate a realistic schedule. In order to be successful, the team needed to provide as accurate of an estimate as possible, but estimates were being provided by the team having had no prior experience in software estimation and with limited resources. In order to accommodate the types of estimates needed by both management and the software team launch, the team established the following: High Level (30,000-foot) Estimate, Low Level (10,000-foot) Estimate, and Detailed Estimate.

High Level Estimate: This estimate is also referred to as the 30,000-foot estimate or a rough order of magnitude and does not contain details since, at that height, you would not see any details. From a conceptual point of view, management may want to integrate some new capability into the software and needs a not-to-exceed cost estimate. Typically, Technical Interchange Meetings (TIM) are held for the purpose of discussing both a proposed capability (including modification to an existing one) and the general idea of how the new software would function. However,

Report Documentation Page				Form Approved OMB No. 0704-0188	
Public reporting burden for the collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to a penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.					
1. REPORT DATE JUL 2011		2. REPORT TYPE		3. DATES COVERED 00-00-2011 to 00-00-2011	
4. TITLE AND SUBTITLE Software Estimation: Developing an Accurate, Reliable Method				5a. CONTRACT NUMBER	
				5b. GRANT NUMBER	
				5c. PROGRAM ELEMENT NUMBER	
6. AUTHOR(S)				5d. PROJECT NUMBER	
				5e. TASK NUMBER	
				5f. WORK UNIT NUMBER	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Air Warfare Center (NAWCWD),414600D MS 2016,507 E. Corsair Street,China Lake,CA,93555-6110				8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)				10. SPONSOR/MONITOR'S ACRONYM(S)	
				11. SPONSOR/MONITOR'S REPORT NUMBER(S)	
12. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution unlimited					
13. SUPPLEMENTARY NOTES					
14. ABSTRACT From a management perspective, it is essential that software estimates used in a TSP launch are as accurate as possible. Significant growth due to estimation inaccuracy can wreak havoc on a team attempting to stay within cost and schedule while executing its established plan. This article discusses how a software team that uses both proxy-based and size-based estimates is able to accurately plan, launch, and execute on schedule.					
15. SUBJECT TERMS					
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT Same as Report (SAR)	18. NUMBER OF PAGES 7	19a. NAME OF RESPONSIBLE PERSON
a. REPORT unclassified	b. ABSTRACT unclassified	c. THIS PAGE unclassified			

at this level there are no formal requirements; hence, the concept of a 30,000-foot estimate. Why? From 30,000 feet, there is not enough detail to get a clear enough picture of all the areas of code that are affected or needed. This type of estimation is used in order to determine if it is feasible and cost effective to proceed into development. Once the TIMs have occurred, an estimate of this type typically takes a day or two to develop.

Low Level Estimate: Once management has determined that the new or modified capability is worth funding and appears to be within the budget, they may request a more refined (i.e., more accurate) assessment. Additionally, the functionality of the proposed capability may be reduced or increased, depending on the budget available. At this point, there is typically a better understanding of what needs to be done. Level 1 requirements (high level system requirements) may be available, along with view graphs calling out detailed functionality; hence, the concept of a 10,000-foot estimate. Things are a little clearer and better defined.

Detailed Estimate: The detailed estimate is performed prior to, and in preparation for, the TSP Launch. During this phase, the Software Engineer (SWE) that is preparing the estimate works with a Systems Engineer¹ to understand both system and software functionality and to evaluate the requirements. The SWE develops a conceptual design that identifies the initial architectural components. These components are then mapped to development tasks, which are workable sized tasks that are identified as development or maintenance tasks. The information associated with each task is documented in a standardized spreadsheet. A set of spreadsheets will be used to document the estimates for each capability with one spreadsheet per affected subsystem. At this point in the Software Estimation, Level 2 system requirements may be available, as well as data from formal program reviews. This is typically in the form of Critical Design Review or Preliminary Design Review slides and action items. These inputs are taken into consideration, if available. The tasks are then divided among the team so that they may prepare detailed task estimates. These task estimates will be documented in the spreadsheets.

Software Estimating (Proxy-based vs. Size-based)

Early on in H2.0 block development, the team realized that the lifecycle for new software development did not address problems associated with software maintenance. Therefore, a lifecycle for maintenance was developed that did not use size-based estimates but used proxy-based estimates instead. The primary reason for both the new lifecycle and the focus on proxy-based estimates is that the development pattern that is followed for maintenance is not consistent with that for new development. For example, in some cases a significant amount of time must be spent identifying the source of the problem with little time up front spent identifying the fix, followed by a significant amount of time spent verifying and testing the fix. Therefore, the software team decided to use the PSP concept of proxy-based estimation. The proxy sizes and times were adjusted over time based on actual data until it stabilized. It took approximately three years before the team identified the four proxy (size to effort) categories [3]:

PROXY	Estimated Effort
Small	6 hours
Medium	17 hours
Large	35 hours
Extra Large	60 hours

Figure 1: Proxy Size-Estimating Table

These proxy sizes have stood the test of time and have not deviated since the H4.0 block build. Originally, the software team used size-based estimates for all new development efforts and proxy-based estimates for maintenance efforts. But this was later abandoned when the team realized that both types of estimation techniques could be used with either new development or maintenance efforts. Analysis of the team using proxy-based estimates showed that the software team was accurate when estimating small and medium tasks, but the complicated, larger maintenance tasks were more difficult to accurately estimate the level of effort involved [3]. The team has developed two strategies for handling these more complicated tasks: (1) change the estimation type to size-based or (2) break the task up into small- and medium-sized tasks and use the proxy-based method on the resulting tasks.

Software Estimating Tool

In an attempt to improve estimation accuracy for large and extra large tasks, the software team developed an estimation tool to assist developers in making proxy-based estimations. At first, the software team felt that this tool was a good concept, but after using the tool for several years, the software team found more disadvantages than advantages. The advantage was that the tool provided new SWEs with a means to ensure that they did not underestimate the size of a task. The disadvantages were experienced by the seasoned SWEs. Once seasoned engineers enter their data, they would often find their engineering judgment disagreeing with the tool. When this was the case, they would simply change the answers to the questions until the tool produced what they felt was the proper proxy size or ignore the proxy size that the tool provided altogether and submit their own. Another disadvantage was that it became difficult to identify what the correct questions for the tool to ask should be, along with the correct computations and weightings to represent each question's impact on the estimated proxy size, to get around the previous disadvantage. This last disadvantage resulted in one SWE spending a considerable portion of time working on refining the size estimation tool rather than working on actual software tasking (i.e., modifying the size estimation tool had become a time-consuming, never-ending chore)[3]. For this reason, the team abandoned the tool concept and adopted establishing an estimating process and spreadsheets to capture the estimates.

A Detailed Estimating Tool is Born

As mentioned earlier, the software team needed to establish a stable way of performing estimations. Initially, estimates were captured in a text file, but this became hard to track and each estimation file did not resemble the next. The team then adopted a spreadsheet approach. At first, the spreadsheet files were simple, but over time they have evolved into MACRO-driven and organized sheets that are very effective in capturing all tasking, size, and lifecycle model information needed for a TSP Launch. This was a departure from the software estimating tool where the SWE would answer questions and the tool would factor in criteria to determine the estimation. The detailed estimating spreadsheets and their usage is described below:

Rollup Sheet: The first sheet in the file is a rollup of all other sheets that contains each component or task and its associated data (i.e., Source Lines Of Code (SLOC), lifecycle used, sub-component name, etc). All SLOC on the first sheet is rolled up at the top of the page to allow size determination. During a launch, there is typically no need to go further in the file than the rollup sheet for populating the Work Breakdown Structure (WBS) size data. A typical estimation workbook will contain the following:

Assumptions Sheet: This sheet captures any assumptions that are being made which could affect the level of effort needed to complete the tasks within the sheets.

Architecture Sheet: This sheet is used to capture the conceptual design/architecture that the tasking sheets support. Any change to design could cause tasking sheets to be added, modified or removed.

Tasking Sheets: Each component/tasking sheet contains the requirement, the files affected, description of changes to the file and the SLOC count. The SLOC is rolled up and displayed at the top of the sheet. Once all requirements are entered, files, changes, and SLOC are identified, the SWE can then determine and select from a drop-down menu the lifecycle model to be used for this component/task. If the lifecycle supports proxies, then the proxy size is also selected from another drop-down menu. Once all tasking is identified for the sheet, it can be integrated into the rollup sheet.

Once the launch is complete, these tasking sheets contain the detailed effort needed to complete each task and can now be used by the assigned SWE in determining what the assigned tasks in the WBS entail.

The detailed estimate spreadsheets works so well that it is now also used for High Level and Low Level estimating, although very little detail is added on the tasking sheets in these estimates.

Quality

The next step in the estimation process is for the estimates to be inspected. For 30,000-foot and 10,000-foot estimates, the software development task team Lead and software subsystem technical expert will review the estimate. For detailed estimates created before a launch and during the development cycle, the software team will review them as part of the final check. During these reviews, all defects including both substantive and minor documentation issues are addressed. All identified defects are reworked as required.

Estimation Currency

As mentioned previously, each capability that is produced in a block development undergoes several iterations of estimates. Initially, in order to support the customer's build decision the software team will create a 30,000-foot estimate. Later, when the customer has made the decision to build the capability, the software team will create a 10,000-foot estimate to support the customer's budgeting and funding activities. These course estimates may be updated as required by the customer. Then, before the first launch to support block development, the software team will create a detailed estimate. This estimate will support the launch activities and will result in a schedule and cost that management and the software team will work with going forward. The software team uses Process Dashboard² to track the development effort. It is this detailed estimate from the launch that will be used as the plan of record in Process Dashboard.

As the development proceeds, new system and software requirements will be added to the project that will require the plan to be modified. The estimates that are associated with these new requirements will be updated, as will the plan of record in Process Dashboard. Also, every six months the team revisits its capability estimates and re-launches the project. This is primarily a realignment of the team's plans to accommodate project progress and changes to the organization's direction and priorities [2]. In order to realign the project plans to the new guidelines, the team must make adjustments for requirements growth and also accommodate the addition and removal of capabilities. The result is that management has current information on the plans for completing the current block. Because the team is continually updating the task completion information in Process Dashboard, management has good quality information on the performance of the team against the plan.

Proof is in the Numbers

So how successful is this approach? Peter Russo, general manager for Microsoft's IT application architecture group comments that:

"There are two fundamental issues in most IT organizations today, one being the ability to accurately predict a project schedule, and the other being the quality of the product once you are finally done" [4].

As Russo points out, identifying a realistic and reliable schedule is essential. This, of course, cannot be done unless you have solid tasking estimates from which to create it. In addition, what is the point of meeting a schedule if the quality of the product is poor? These issues transcend the boundaries of just an IT organization and apply to any organization developing software on a timeline within a fixed budget. Figure 2 shows the actual size in SLOC of the effort for blocks H4.0 – H6.0. Note that the source size grew 46K between builds H4.0 and H5.0 and 35K between H5.0 and H6.0. SLOC size is determined by the number of SLOC that are added and modified to the existing baseline.

Block	Size (SLOC)
H4.0	47,997
H5.0	94,682
H6.0	129,607

Figure 2: Actual Size by block

Product Quality: There are different measures to indicate the quality in a product. The book, *Code Complete*, indicates that the Industry Average defect density is between 15 and 50 defects per 1,000 Lines of Code (KLOC). Microsoft applications are produced with a defect density of about 0.5 defects per KLOC in released code. Organizations using the Harlan Mills pioneered “cleanroom development” technique have been able to achieve densities as low as 0.1 defects per KLOC [5]. The software team uses defect density (defects per KLOC) to determine the quality of its products. Figure 3 compares the defect density of each block delivery against defects delivered by CMM® level 1 and level 5 organizations. Defects identified here for CMM level 1 and level 5 are captured from Capers Jones who has identified software delivered defects at each level of the SEI CMM [6]. As can be seen, the defect density for all blocks is significantly lower than that expected of a CMM level 5 organization. In addition, the quality is better than Microsoft’s threshold and approaching that expected by those using the cleanroom development technique.

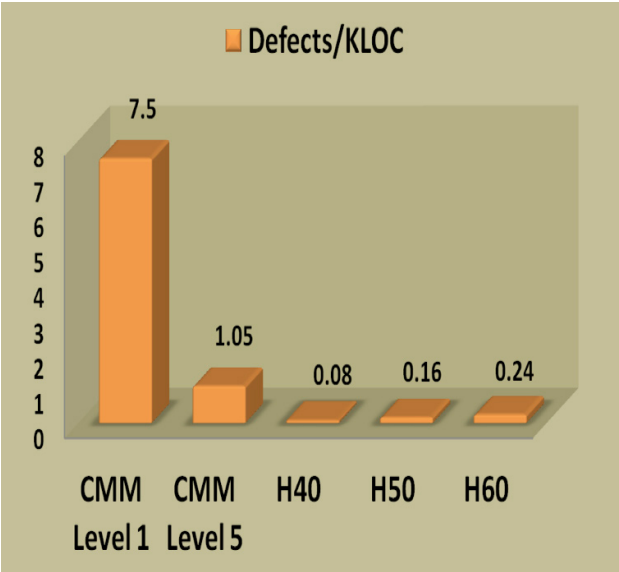


Figure 3: Defect Density

One notable trend is that the quality of the finished product decreases as the size of the product increases. For these three data points the relationship is almost linear; the defect density increases by about 0.002 per KLOC. Other factors including task complexity and team volatility may have an affect on the quality, but were not factored into the data. That being said, the quality of the software at release is high.

Proxy Estimating Accuracy: As mentioned earlier, establishing a reliable schedule requires accurate software estimates. Given that the team is developing high quality products, figures 4-7 illustrate how well the team did at estimating task sizes indicated in the Proxy Size Estimating Table (Figure 1).

For small software development tasks (Figure 4), the software team did an excellent job identifying them and improved its estimation accuracy with each consecutive development effort.

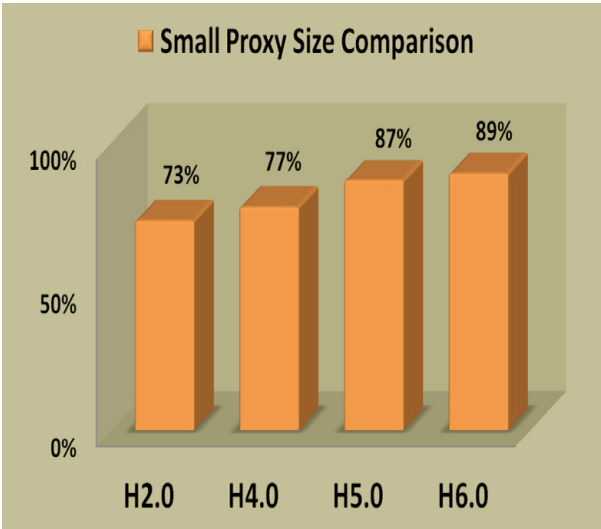


Figure 4: Small Proxy Estimation Accuracy

The number of tasks identified for each block was 11, 62, 45, and 204 respectively. Although the number of tasks grew significantly by H6.0, the team was still able to accurately estimate this size category.

For medium software development efforts (Figure 5), the software team did a good job of identifying these tasks. They improved with each development effort given that H5.0 and H6.0 only varied by 2%. The number of tasks identified for each block

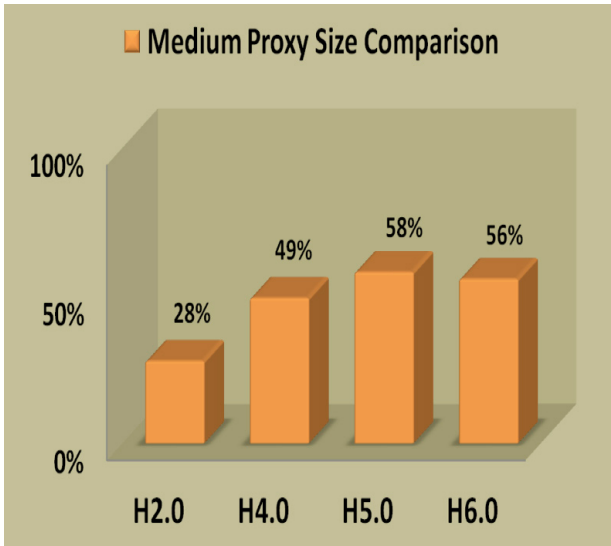


Figure 5: Medium Proxy Estimation Accuracy

was 61, 65, 50, and 291 respectively. Although the number of tasks grew significantly for H6.0, the team was still reasonably accurate in estimating this size category.

For large software development efforts (Figure 6), the number of tasks identified for each block was 37, 29, and 76 respectively. H5.0 did not have enough data points in this proxy category for comparison. Here the team did a good job of estimating and is improving in this area. But because tasks of this size tend to

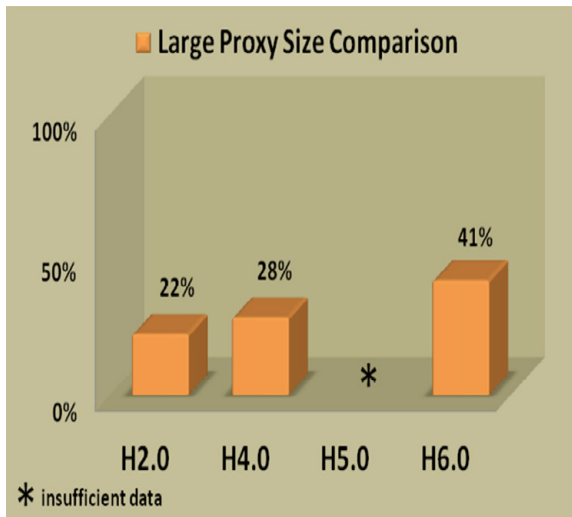


Figure 6: Large Proxy Estimation Accuracy

be more complex, it is more difficult to estimate as accurately as compared to smaller task sizes.

For very large software development efforts, the number of tasks identified for H2.0 was 18 and H6.0 was 21. Both H4.0 and H5.0 did not have enough data points in this proxy category for comparison. As one would expect, tasks this size are significantly more complex and difficult to estimate. In some cases,

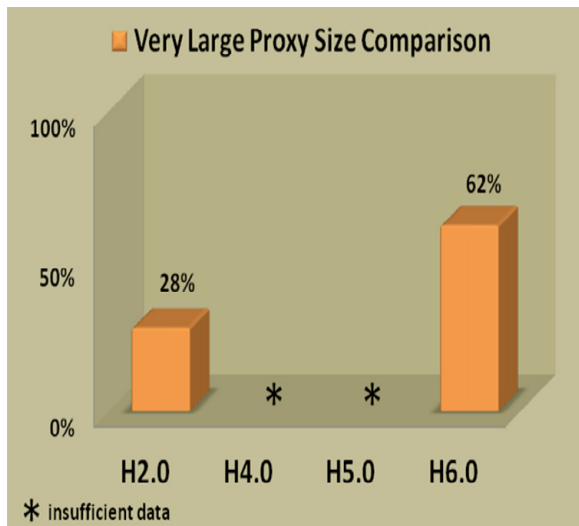


Figure 7: Very Large Proxy Estimation Accuracy

these tasks can be more of what is referred to as “science projects,” where the task is known to be very complicated and has too many unknowns to determine what is required. The team has gotten better at breaking down complex tasks into multiple smaller tasks. Overall, the team trend appears to be getting better at identifying tasks of this size.

In summary, for proxy-based estimations, the software team did an excellent job estimating the number of small tasks, but as the data indicates, as the tasks became larger and more complex it became more difficult to estimate the level of effort

involved. So how well does this estimation methodology support the production of an accurate plan?

Plan Accuracy: Progress against the plan is described in terms of earned value, which is based upon the estimated labor hours needed to complete each task. As the team completes tasks, they are able to determine how well they have done in meeting the plan. Figures 8-11 show how well the team's execution (earned value) compared to the plan (planned value) for blocks H2.0-H6.0.

Figure 8 illustrates the planned versus actual earned value for the H2.0 block project. Initially the actual earned value was accrued at a significantly higher rate than the planned earned

H2.0 EV Chart

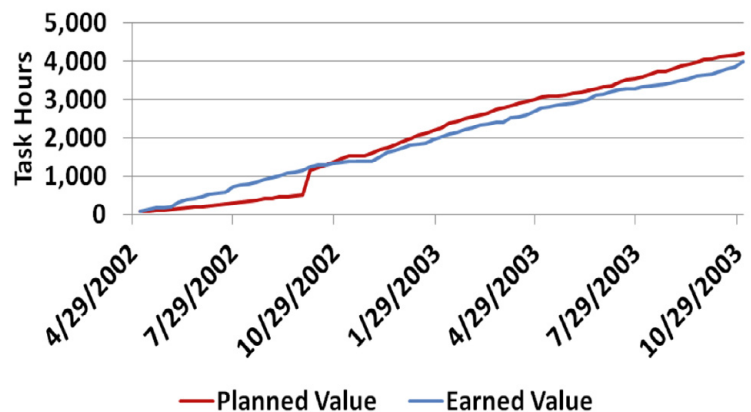


Figure 8: H2.0 Schedule Accuracy

value. This was a result of the team overestimating the H2.0 tasking efforts. At this point, the team had not yet established a reliable estimating method. A relaunch occurred where the graph of the planned value abruptly joins the actual earned value curve (October 2002). After this relaunch, the team accrued earned value more closely to the planned earned value.

The planned versus actual earned value for the H4.0 block project is shown in Figure 9. Between August and November of 2004, the graph is flat due to missing project data. The team at this point had established the estimating sheets but still had not bridged

H4.0 EV Chart

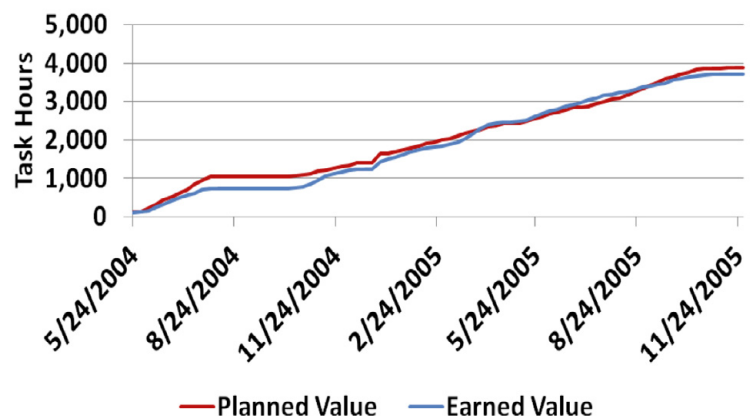


Figure 9: H4.0 Schedule Accuracy

the gap between low level and detailed estimates. Although the software team's accrued earned value followed the planned earned value relatively closely, there are numerous steep and shallow slopes of the earned value line, reflecting periods during which the team received extra credit for completing over-estimated tasks, or too little credit for completing under-estimated tasks.

By the time the software team had launched H5.0, the estimating method was fully established. Figure 10 illustrates the planned versus actual earned value for the H5.0 block project. The separation of planned versus actual earned value in the latter

H5.0 EV Chart

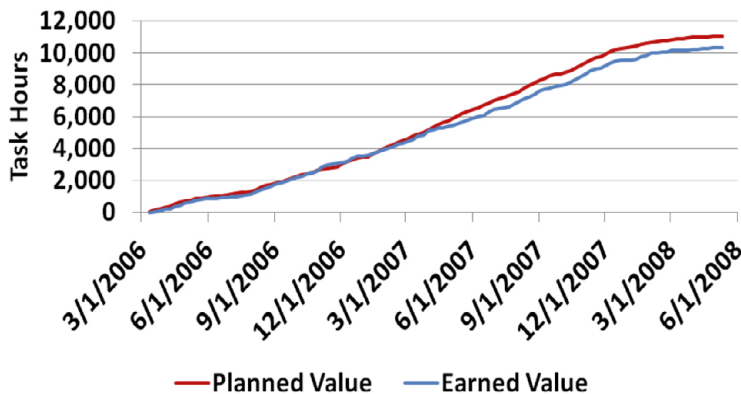


Figure 10: H5.0 Schedule Accuracy

half of the project is due to the delay of several tasks that were not related directly to the product development. These efforts include non-product documentation, post-mortem data analysis, and other non-block related tasks. The team now had an established reliable estimating methodology and it was beginning to show.

Under the H6.0 development effort, the team continued to refine its estimating process. The planned versus actual earned value chart is shown in Figure 11. For 31 months, the software team was able to accrue earned value very consistently with the expected planned value. Although it had taken several blocks, this is the type of planning and execution that the team had hoped for and had finally achieved.

H6.0 EV Chart

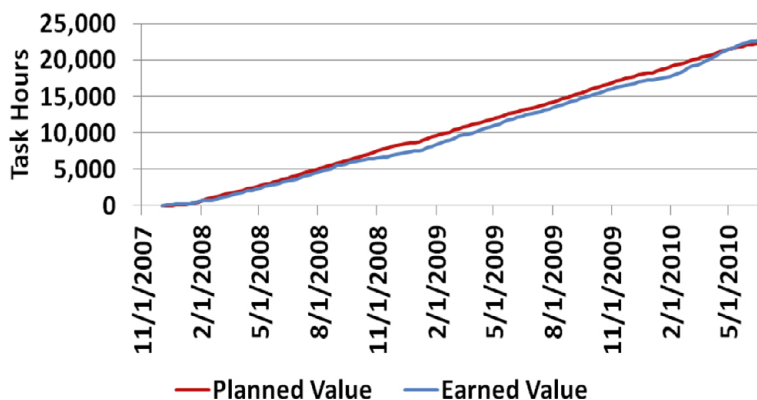


Figure 11: H6.0 Schedule Accuracy

Summary

The team's approach in estimating has enabled it to produce a realistic plan that the team, its customers, and its management are able to effectively use. Even though the team is now able to accurately produce a plan from established estimations, it continues to look for ways to improve its estimating ability because, in the end, it all begins with quality estimates. ♦

Disclaimer:

®CMM is registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

SMPSP and SMTSP are service marks of Carnegie Mellon University.

REFERENCES

1. Rickets, Chris A, "A TSP Software Maintenance Life Cycle", CrossTalk, March, 2005.
2. Koch, Alan S, "TSP Can Be the Building blocks for CMMI", CrossTalk, March, 2005.
3. Hodgins, Brad, Rickets, Chris, Lindeman, Robert "How TSP Implementation Has Evolved at AV-8B," TSP Symposium September 2007.
4. Grojean, Carol A, "Microsoft's IT Organization Uses PSP/TSP to Achieve Engineering Excellence", CrossTalk, March, 2005.
5. McConnell, Steve, "CODE Complete", Microsoft Press, 2004.
6. Jones, Capers, "Software Assessments, Benchmarks, and Best Practices", Addison-Wesley Professional, April 2000.

NOTES

1. At the AV-8B Joint System Support Activity, the systems engineering team is responsible for system and software requirements.
2. Process Dashboard is a software planning and tracking tool by Tuma Solutions, LLC.

ABOUT THE AUTHORS



Robert Sinclair is a senior engineer at the Naval Air Warfare Center in China Lake, Ca. He has more than 20 years of experience in developing software in various capacities for the Air Force and the Navy. Mr. Sinclair is currently the supervisor of a group that is developing embedded software. He holds a bachelor's degree in Electrical Engineering from Iowa State University and a Master's in System Engineering from Virginia Tech.

Robert Sinclair
NAWCWD
414600D MS 2016
507 E. Corsair Street
China Lake, CA 93555-6110
760-939-6989



Chris Ricketts is a senior engineer at the Naval Air Warfare Center, China Lake, Ca. He has more than 26 years of experience in developing embedded software for the Navy. Mr. Ricketts has been involved in Harrier block upgrades H2.0 – H6.0 and is currently the Mission Systems Computer Technical Lead for the H6.1 block effort and the supervisor for the software developing team. Mr. Ricketts holds both a BS and MS in Computer Science from California State University, Chico.

Chris Ricketts
NAWCWD
414300D MS 2016
507 E. Corsair St.
China Lake, CA 93555-6100
Phone: 760-939-5838



Brad Hodgins is an interim TSP Mentor Coach, SEI-Authorized TSP Coach, SEI-Certified PSP/TSP Instructor, and SEI-Certified Software Developer for the Process Resource Team of the Engineering Division of the Naval Air Systems Command (NAVAIR) located at China Lake, California. Hodgins has been with NAVAIR for 27 years. He has over 20 years experience as a software engineer developing simulation and avionics software. He has been applying PSP/TSP for over eleven years and has coached over 50 TSP/TPI launches. Hodgins earned a BS in Computer Science from California State University, Chico.

Brad Hodgins
NAWCWD
414600 MS 6308
9100 N. Knox Road, Bldg. 01494
China Lake, CA 93555-6110
Phone: 760-939-0666

WANTED

Electrical Engineers and Computer Scientists *Be on the Cutting Edge of Software Development*

The Software Maintenance Group at Hill Air Force Base is recruiting **civilian positions** (*U.S. Citizenship Required*). Benefits include paid vacation, health care plans, matching retirement fund, tuition assistance and time off for fitness activities. **Become part of the best and brightest!**

Hill Air Force Base is located close to the Wasatch and Uinta mountains with many recreational opportunities available.



Send resumes to:
phil.coumans@hill.af.mil
or call (801) 586-5325

Visit us at:
<http://www.309SMXG.hill.af.mil>

