

COCOMO II Predicts the Effects of Collaborative Software Development and Sustaining Engineering

David K. Mann, Ph.D., PMP, CM
United Space Alliance (USA)
Launch Processing System Engineering
8550 Astronaut Boulevard, MS: USK-646
Cape Canaveral, FL 32920-4304
(321) 861-7234

Abstract

This paper explores the use of the COCOMO II model to predict results of applying collaborative software development and sustaining engineering to aerospace and government software development projects. Empirical data from the Space Shuttle Telemetry Ground Station upgrade, recently completed using collaborative software development, is compared to model predictions. Extrapolation is done to explore possible benefits of the method to other upgrade or development projects.

This new software development method resulted in several positive effects on the ground station upgrade project over the standard method ordinarily employed to develop and upgrade systems. Prior to implementing this new method, the upgrade project had been under development for several years using the standard development method. The standard software development method involves developing custom applications using COTS hardware, operating systems, development tools, drivers and compilers. The new method involved adopting a commercial telemetry ground station software package to provide basic functionality and building additional required capabilities in a collaborative manner with the COTS software. Positive effects recounted in the paper include; a reduction in the initial software development costs, a reduction in the software time to market, improved marketability of the software technology developed, improved product quality, and improved maintainability. Two reasons the method was so successful are explored using the COCOMO II model. The first reason explored is that by employing this collaborative method we changed the fundamental skill mix of the software development team. Initially the team was expert in the problem domain and weak in the programming and information technology areas. After changing the development method the skills of the software development team reflected a balanced expert team. The second reason the method was so successful is that the project effectively leveraged existing code and architecture of the commercial product. The paper explores possible application of this development methodology to other government and NASA programs and the probable benefits.

Introduction

Daniel Golden, NASA Administrator, in his strategic outlook for 1999 (reference: <http://hq.nasa.gov/office/nsp/outlook.htm>) provides a statement of strategic intent for the agency. In this statement he outlines a three-part mission in which "Technology Development and Transfer" is a cornerstone of the mission for the agency in the coming year. This is consistent with the 1999 external assessment (reference: <http://hq.nasa.gov/office/nsp/assess.htm>) in which the Administration places priority on the promotion of "high technology for economic growth through effective partnerships". The prime Space Flight Operations Contract, SFOC (reference:

Contract NAS 9-20000) awarded to United Space Alliance, section G-14 and G-15 requires the contractor to provide a portion of the contract funds to small business and to support the "Government's Technology Transfer Program". This new development method is an example of how USA is exploring new ways to increase the marketability of technology developed on the space program while decreasing Space Shuttle Program development and operating costs.

This new collaborative development method involves adopting a software application counterpart available in industry to provide baseline functionality and developing additional capabilities required for the upgrade or replacement system in collaboration with the software vendor. The application of this new development method more effectively leveraged technology and expertise available in industry to reduce initial software acquisition cost and time to market while providing a superior product to Space Shuttle operations. In addition, the technology developed is built on the state-of-the-art rather than reinventing the state-of-the-art, making the technology developed more valuable to industry and the American public.

The project defined a software development turnaround in terms of key project management metrics (i.e. cost, schedule and technical). Turnaround was defined as a 50% reduction in anticipated software development labor and time to market. The improvement in the technical merit was a little harder to quantify. Two surveys were performed as part of a comparative analysis. Three key technical categories were defined and a turnaround was defined as a marked improvement in two of the three. The categories were marketability, software product quality and maintainability. Attributes were defined within each category. The first survey was designed to determine the importance of the attributes within each category. The second survey was performed after software development was complete and scored the delivered product against the most likely outcome of continued development on the custom code. The results of these surveys were organized and presented in a Kepner Tregoe decision matrix for comparative analysis. A marked improvement was defined as 100% improvement in absolute score within a category.

Development Method Genesis And Vendor Selection Process

A detailed estimate to complete the project assuming continued status quo software development was performed, which provides a baseline for evaluating, improved project performance after the change in direction. This estimate to complete was based on a functional analysis of the custom code under development against the functional requirements for the upgrade. Functionality was divided into three categories for the purpose of this analysis; telemetric, station management, and data products and tools. Telemetric functionality was defined as the capabilities to acquire measurement data from the PCM or FM carriers, route this data to a location on the ground station, and capture the data in a file for production of data products. Station management functionality was defined as the capability to setup and manage, data acquisition as well as monitor real time elements status. Data product and tools were defined as the capability to process the raw data acquired into data products. Figure 1 is a summary of this analysis. The entire pie in each category represents the functionality required for the upgrade in each category. The three slivers contained in the regions labeled "Functionality Addressed in the Custom Software" represents capability outlined in the custom code. The two black slivers in this region represent the capabilities inherent in the first two releases of the custom code. The third shaded slice in this region represents capabilities that were outlined but not functional. The slices remaining outside the region represents capabilities requiring new development work to deliver. The analysis revealed that approximately one third of the telemetric, three quarters of the station

management and two thirds of the data products and tools capabilities required new development assuming continued status quo software development. Project leadership and management derived an estimate to complete for the custom software. This estimate was based on a detailed knowledge of the functionality required (function point analysis), performance histories of the developers involved and developer's estimates. It was estimated that to complete the custom code would require an additional twenty man-years of productive software development effort expended over five years with a probable growth of twenty percent in both manpower and schedule.

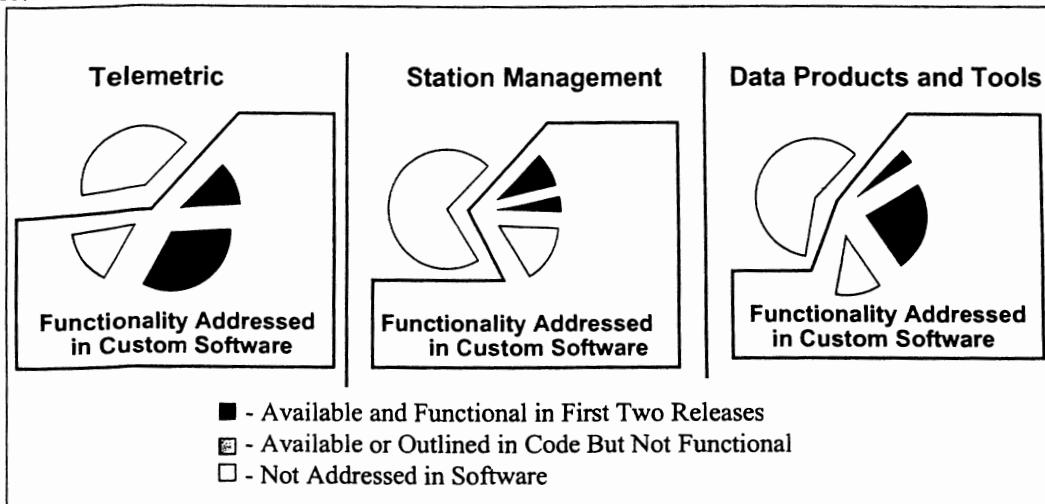


Figure 1 *Evaluation of the Custom Code Revealed Large Gaps in Functionality*

An industry survey coupled with a series of product evaluations were performed to determine if a Commercial Off The Shelf (COTS) product was available that met the functional requirements for the upgrade. The survey revealed that no commercially available software package met the requirements of the upgrade. There were however several close functional counterparts, Veda Systems, Huerikon, Harris, Metraplex, APLabs Inc., Avtec and Acromag. Hardware for the upgrade project had already been purchased and major change to the hardware architecture was cost prohibitive. As product and vendor investigations progressed, it became apparent that general-purpose telemetry software packages were coupled to particular hardware architectures. Of course, anyone can do a port, but the challenges associated with developing additional capabilities concurrent with a port to a new hardware architecture was an approach deemed to have excessive risk. In addition, the vendor selected had to provide flexibility in development and licensing that will be discussed later. These factors lead to a single viable vendor and product combination. APLabs Inc. wrote VMEwindow to a hardware architecture very similar to the one that was in storage for the upgrade and provided the flexibility to complete the project. A functionality assessment, similar to the one performed above, was performed in order to scope the software development effort required and justify the change in strategic direction to management.

Collaborative Software Development

Collaborative software development started the second week of March 1998. The first order of business was to train USA developers in the VMEwindow development environment. Systems engineering was performed to identify the system level requirements and identify gaps in

functionality between the baseline VMEwindow product and the required upgrade ground station. Although there were many minor modifications, the collaboration resulted in 4 software products, with innovators from USA and APLabs Inc. These were submitted to NASA for Technology Transfer. Software development was completed the second week in November 1998. These software products represent modification to existing technology to improve and expand the capabilities of the baseline product. Figure 3 is a block diagram of the upgrade ground station architecture and is referenced in the following discussions on the software developed. The hardware architecture consists of Sun workstations connected to multiple PCM and FM processing VME chassis which use reflective memory as a real time data transport mechanism. The chassis use Motorola processors and SBS Berg telemetry System cards. The software architecture consists of the VxWorks operating system, VMEwindow, Matlab, Dataviews, PVWave. The software products developed for the project and submitted to NASA for technology transfer are described below.

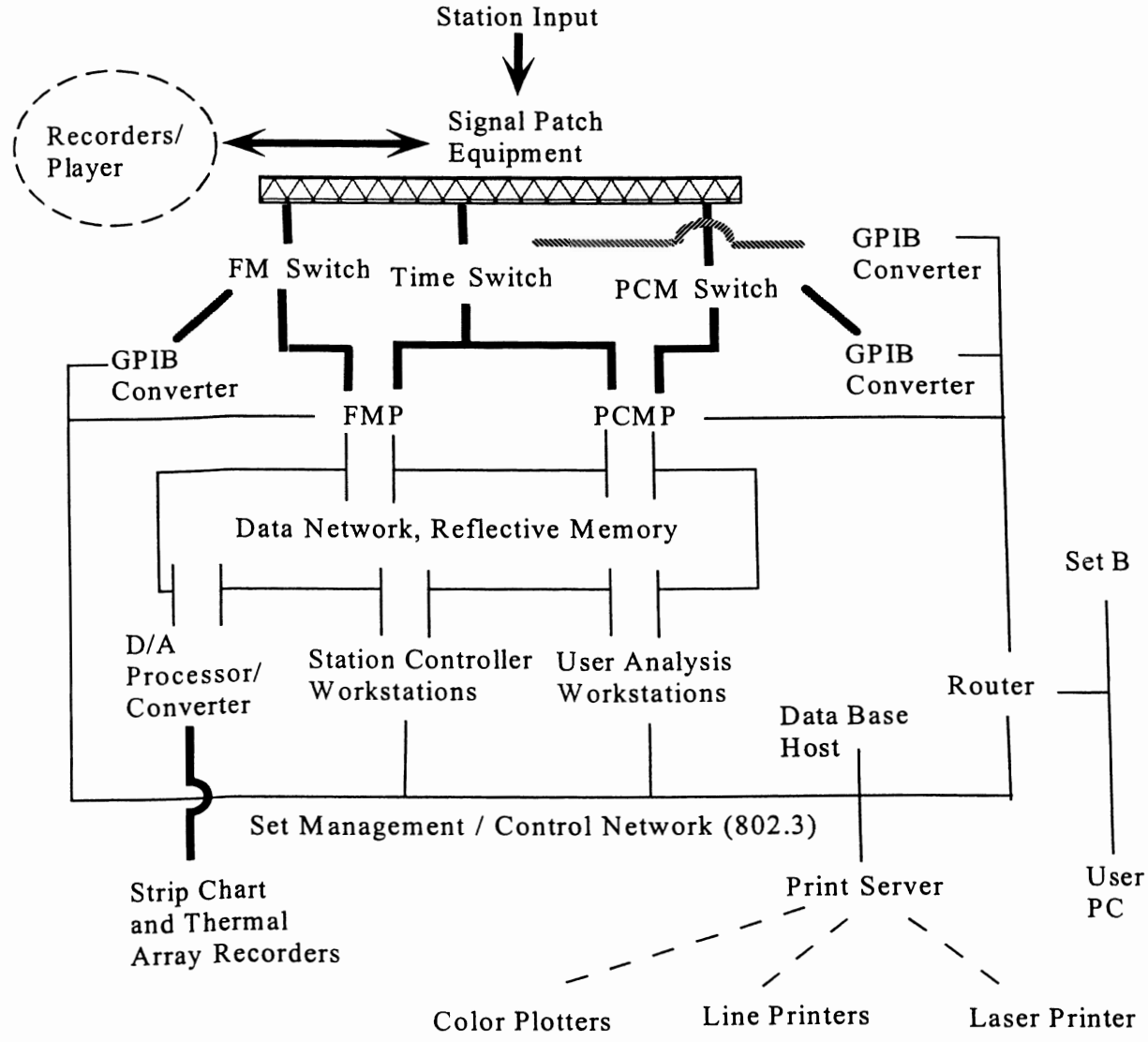


Figure 3 Ground Station Overview
Sbus Scramnet Interface for VMEwindow

A reflective memory network shown in the center of Figure 3 is the primary data transport vehicle for raw data from the PCM Processors (PCMPs) to the Digital to Analog Programmable Converters, User and Station Controller Workstations. The VMEwindow ground station telemetry package did not support an SBus SCRAMnet interface required to get data to the workstations. The challenge was to develop an application using the VMEwindow development environment that would provide the capability to acquire and display data at a single SBus workstation from multiple real time telemetry processors in a deterministic manner with minimal latency. The code needed to be integrated within the VMEwindow environment providing the operator with a consistent interface. It was decided that a derivative work could be produced from the code already available for interface to the VME version of the SCRAMnet card. It was negotiated that APLabs would perform code modification and initial development testing and USA would provide the reflective memory topology, memory offset and data requirements in addition to concurrent code review, integration, testing.

Stream Definition Flat File Import Capability for VMEwindows

This software provides the capability to automatically load telemetry stream definition. This capability was seen as critical to the design because of the thousands of measurements that must be loaded to support each mission. Space Shuttle PCM down link, data location and unit information is contained in a data base. Baseline VMEwindow provides a manual utility to load this information into a stream definition but did not provide the capability to automatically load stream definition. The software product developed provides the capability to create an ASCII flat file from an SQL data base and import this flat file into the VMEwindows stream definition. This capability not only reduces time required to setup to support a mission but improves the reliability of the software load by cutting down on input errors. The user is prompted for parameters to define a PCM stream and accepts lists of measurement names read from an existing file and accesses the data base to create an ASCII flat file. The generated file contains all of the information necessary to define PCM stream and populate the stream definition in VMEwindows.

GPIB Board Setup and Control for FM Snapshots and Calibration

This software provides the capability to setup and control the NI1014 General Purpose Interface Bus (GPIB) board for specialized Frequency Modulation (FM) functions. Although a generic GPIB interface capability was available in the baseline VMEwindow environment, it did not meet the requirements of the project. The system uses a Metraplex digital discriminator and a Keithly switcher. The setup and control software allows individual control of both the discriminator and the switch and automated control of report generation. The setup functions were integrated into the ground station software as a newly developed VMEwindow icon. Board level control is provided using existing APLabs Inc., driver software. The software provides a calibration and FM Snapshot capabilities. A FM test signal that shifts the frequency over the bandwidth to represent five levels between -5V to 5V is input to the digital discriminator. The calibration software provides an average of 5 samples at each of these data levels. Once the set is calibrated, a FM snapshot can be produced providing the average, minimum, and maximum values for a 100 samples of real time data.

Datel 622 Digital to Analog Driver, Setup, and Control

This software product provides the ability to produce thermal array charts of data with loss of signal event indicators. The driver and control software provides the capability to setup and control the Digital to Analog (D/A) conversion functions of the board. This capability was not available in the baseline VMEwindow environment. The setup functions were integrated into the ground station software as a newly developed VMEwindow icon. The control software provides the capability to select data channels to be output, select event indicators for edge trace output, scale processing, and calibration functions.

Collaborative Support, Development, and Licensing Agreement

There are several challenges facing successful application of a collaborative software development on a project in support of Shuttle operations. Some challenges stem from the perception that the ability to support operations is somehow compromised, while others from the notion that derivative or new software technology developed adjunct to a baseline application is not transferable to NASA and industry. Another argument in favor of in house custom software development is that the developers are often the most qualified to troubleshoot and repair time critical bugs during processing. Additionally, if the vendor were to go out of business or drop the product the space program would run the risk of losing support for operational software. The project had to ensure that each of these issues were addressed and an equal or superior capability would be available after the upgrade was complete.

These issues were addressed in several ways. First, in-house software developers were trained and certified as VMEwindow developers. This provided us the capability to collaborate on the development of required new capabilities and sustain the software after the upgrade was complete. Second, we negotiated a "Collaborative Support, Development and Licensing Agreement" with the vendor. This agreement has several clauses designed to address the concerns above. Rights to the source code for the purposes of RPS Shuttle processing and technology transfer to NASA were secured in a "Project Buyout License" clause. This eliminated the need to escrow source code with a third party and enabled ad hoc modification of the baseline product during mission support without infringement concerns. Acquiring special software developer support, ensuring that immediate attention is paid to software issues during critical processing times in collaboration with in-house developers, addressed operational support concerns. A product upgrade cycle is defined in the agreement where new version software is provided along with user documentation and installation assistance. This ensures that we have the ability to stay current while minimizing configuration management costs. The software development collaborations were so successful that special provisions were negotiated to ensure future endeavors would adopt a similar course. Subsequently, two new software innovations have been developed and will be written up for technology transfer.

Empirical Findings

Cost and Schedule

Comparing the actual resources expended to the estimated resources required to complete the custom code (i.e., assuming continued status quo development) provides software development cost avoidance. The cost associated with purchasing the COTS software and development services is conservatively converted to equivalent manpower and added to the actual in house labor expended to complete the software. The resulting figure is 6.6 man-years. When

compared to the estimate to complete the custom code reveals 13.4 to 17.4 man-years software development labor avoidance. Converting this to equivalent dollars reveals more than \$1M in cost avoidance. Table 1 presents a summary of project costs and an estimate of the manpower expended to produce the code. Development services were itemized on the contract with APLabs. These services were figured at 37 days of technical support at \$1000 per day. Converting this figure to man-hours effort using a representative labor rate of \$47 per hour we get 787 man-hours. We now add this to the actual USA software development labor charged to arrive at an approximate equivalent manpower expended of 3.2 man-years. Table 2 presents a summary of the calculations used to estimate the reduction in effort as a percentage of the total effort. The estimated effort to complete the custom code derived during the system functional analysis is used in conjunction with equivalent manpower expended to derive the estimated labor avoidance. This resulting labor avoidance can then be used to calculate percentage avoidance as a function of the total estimated software development labor effort. The high and low values are averaged to arrive at an estimated 85 percent reduction in software development effort. Non-Labor additional expenditures incurred to acquire the COTS baseline product and additional hardware components and upgrades represent the penalty paid to pursue this new development avenue. When we convert these labor avoidance estimated to cost we can figure the penalty as a percentage of the total cost avoidance. An approximate net avoidance can then be figured by subtracting this penalty from the percent total labor avoidance. This results in a 64 percent net estimated avoidance.

Software development authority to proceed was given in January 1998 and software development was complete the second week in November 1998. This meant that the software development actually took eight months rather than the five to six years estimated to complete the custom code. This represents an 86 to 89% reduction in the anticipated software development time to market.

Table 1 Project Cost Worksheet

| | Jan | Feb | Mar | Apr | May | Jun | Jul | Aug | Sep | Oct | Nov | Total |
|-------------------------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|--------|-------|
| A007212KM | 54 | 33 | 6 | 5 | | | 410 | 703 | 787 | 790 | 700 | |
| A007212K1 | 91 | 129 | 192 | 186 | 597 | 405 | 174 | | | | | |
| Mhours/month | 145 | 162 | 198 | 191 | 597 | 405 | 584 | 703 | 787 | 790 | 700 | 5262 |
| | | | | | | | | | | | Myears | 2.7 |
| Technical Support Contract \$ | | | | | | | | | | | | 37000 |
| Labor Rate \$/H | | | | | | | | | | | | 47 |
| Technical Support Hours | | | | | | | | | | | | 787 |
| Equivelant Hours | | | | | | | | | | | | 6049 |
| Equivelant ManYears | | | | | | | | | | | | 3.2 |

Table 2 Percent Avoidance

| | Low | High | Average |
|---------------------------|---------|---------|---------|
| Labor ETC | 20 | 24 | |
| Equivalent Man Years | 3.2 | 3.2 | |
| Labor Avoidance | 16.8 | 20.8 | |
| Cost per Man-year | 90000 | 90000 | |
| Project projected cost | 1800000 | 2160000 | |
| Cost Avoidance | 1512000 | 1872000 | |
| % Labor Avoidance | 84% | 87% | 85% |
| Non Labor Penalty | 360000 | 360000 | |
| Penalty as % of Avoidance | 24% | 19% | 22% |
| Net Total Avoidance | 60% | 67% | 64% |

Technical Merit

A comparative analysis designed to determine the relative technical merit of the delivered software to the most probable product assuming continued work on the custom code was performed. Expert engineering judgment is relied upon as the basis for this analysis through two surveys. The first column of table 1 lists the categories defined to represent technical merit for the purposes of this analysis. Sub-indentures under each of the three categories (i.e. Marketability, Software Product Quality and Maintainability) are attributes of the categories. All responses were normalized, averaged and reported on a 1 to 10 scale where 10 was the most important or best. The second column represents the results of the first survey. This survey was designed to determine the relative importance of the attributes within a category from the perspective of the management. The third and fourth columns represent the results from the second survey. The respondents scored the delivered software and the most probable outcome of continued status quo development. The additional costs associated with the procurement of the COTS software and development (i.e.. penalty) was converted to equivalent manpower and the respondents were asked to estimate assuming these additional resources were brought to bear on the custom software development. The fifth and sixth columns list the absolute scores for the custom and delivered product respectively. These figures are totaled for each category to obtain an absolute relative score for each of the alternative methods. Comparing absolute scores reveals a 3 to 4 fold improvement in absolute score in every category.

Table 1 Method Comparison Matrix

| 1 | 2 | 3 | 4 | 5 | 6 |
|-----------------------------------|---------------------------|---------------------------------------|--|--|--------------------------------------|
| | Average Normalized Weight | Average Probable Custom Product Score | Average Software Product Delivered Score | Absolute Score Probable Custom Product | Absolute Score of Software Delivered |
| Marketability | | | | | |
| Value to Current Customer | 7.78 | 1.67 | 9.33 | 13 | 73 |
| Value to Other NASA Projects | 4.70 | 2.50 | 8.50 | 12 | 40 |
| Value to Industry | 2.22 | 2.67 | 8.33 | 6 | 18 |
| Value to Future Shuttle Customers | 6.38 | 2.00 | 9.00 | 13 | 57 |
| Value to American Public | 3.92 | 4.17 | 6.83 | 16 | 27 |
| Total = | | | | 60 | 215 |
| Software Product Quality | | | | | |
| Usability | 5.07 | 2.67 | 8.33 | 14 | 42 |
| Reliability | 8.26 | 1.83 | 9.17 | 15 | 76 |
| Functionality and Versatility | 4.57 | 1.83 | 9.17 | 8 | 42 |
| Extensibility | 2.09 | 1.50 | 9.50 | 3 | 20 |
| Total = | | | | 40 | 180 |
| Maintainability | | | | | |
| Training Programs | 5.24 | 2.33 | 8.67 | 12 | 45 |
| Availability of Trained Personnel | 3.69 | 2.83 | 8.17 | 10 | 30 |
| Ability to Enhance Software | 5.55 | 1.67 | 9.33 | 9 | 52 |
| Documentation | 3.42 | 2.00 | 9.00 | 7 | 31 |
| Configuration Management | 7.09 | 3.58 | 7.42 | 25 | 53 |
| Total = | | | | 64 | 211 |

COCOMO II Predictions

Intuitively, the development team thought that there were two main contributors to the dramatic cost avoidance estimates shown above. First, the fundamental skill mix of the development team was changed by the addition of the APLabs personnel. Secondly, we effectively leveraged the commercial product and development tools. In an attempt to affirm these intuitive assertions we downloaded COCOMO II 1999 Model from the University of Southern California. Graduate student programming teams under the leadership of Dr. Ellis Horowitz, Director, Distance Education and Information Technology Programs University of Southern California, developed this software-estimating tool. Dr. Barry Boehm first published the Original COCOMO model in 1981. It was thought that a correlation between the model output accounting for the skill mix and adaptation over similar lines of code (LOC) estimates with empirical findings would affirm these assertions. Interviews were performed with project and program management and software developers to arrive at consensus input to the model. Where consensus was not achievable majority responses were used. Definitions and explanations were read to the interviewees from the COCOMO II Users Manual downloaded with the application. The participants are listed under acknowledgements.

Development Team Skills

Table 3 presents a summary of the output from the COCOMO II model. The table is divided into three blocks. The first block is a summary of the model output compensating for the change in skills of the software development team. The second block presents model output compensating for adaptation or leveraging of the commercial code and development environment. The final block presents a composite of the two effects.

The composite development team thought that the original USA team was strong in problem domain expertise (i.e. Space Shuttle telemetry, systems and operations) but were weak in real-time platform and large scale Higher Order Language (HOL) programming expertise. The majority of the programmers that were assigned to the ground station development project were very experienced software-sustaining engineers. Most of the code under sustainment was written in Modcomp Assembler, Fortran, C and Ground Oriented Avionics Language (GOAL). The local C code represented a small fraction of the code responsibility. The developers had all been to VxWorks training. Large-scale systems development experience in C or VxWorks was non-existent. On the other hand the developers at APLabs were seen to be weak in problem domain expertise and strong in real-time platform expertise and large scale C development. The APLabs developers had been intimately involved in the development of the VMEWindow and therefore most familiar with the development environment and tools. In keeping with this logic the development team rated the original USA development team "very high" in Analyst Capability (ACAP) and Applications Experience (AEXP) "nominal" in Programmer's Capability (PCAP) and "low" in Platform Experience (PEXP) and Language Expertise (LEXP). The addition of APLabs personnel to the development teams bolstered the ratings in the areas of Programmer's Capability, Platform Experience and Language and Tools for "low" to "high". Personnel Continuity (PCON) and all other parameters were left nominal.

The model was run from 300KLOC to 900KLOC, first assuming the skill ratings of the USA team and then the combined team. At all SLOC settings the model estimated a 42% reduction in software development effort.

Table 3 COCOMO II Model Output

| SLOC | Skills | | | | Adaptation | | | Composite | | |
|--------|--------|--------|--------|--------------------------|--------------------------|--------------------------|--------------------------|------------------------|----------------------------|--------------------------|
| | Nom | Before | After | Percent Effort Reduction | Custom Code Continuation | Leverage VMEWindows Code | Percent Effort Reduction | USA Skills Custom Code | USA/APLabs VMEWindows Code | Percent Effort Reduction |
| | | | | | | | | | | |
| 300000 | 1557.5 | 1064.2 | 612.5 | 42% | 1118.5 | 265.3 | 76% | 764.2 | 126.4 | 83% |
| 400000 | 2137.1 | 1460.3 | 840.4 | 42% | 1534.7 | 364.1 | 76% | 1048.6 | 173.4 | 83% |
| 500000 | 2731.5 | 1866.4 | 1074.1 | 42% | 1961.5 | 465.3 | 76% | 1340.3 | 221.6 | 83% |
| 600000 | 3338 | 2280.7 | 1312.6 | 42% | 2397 | 568.6 | 76% | 1637.8 | 270.8 | 83% |
| 700000 | 3954.6 | 2702.1 | 1555 | 42% | 2839.9 | 673.7 | 76% | 1940.4 | 320.8 | 83% |
| 800000 | 4580.1 | 3129.5 | 1801 | 42% | 3289 | 780.2 | 76% | 2247.3 | 371.6 | 83% |
| 900000 | 5213.5 | 3562.2 | 2050.1 | 42% | 3743.9 | 888.1 | 76% | 2558.1 | 423 | 83% |

In compensating for adaptation, the custom code that had been under development was scored along side that of the VMEWindow product used as a baseline for development efforts. Those members of the software development team that were familiar with the custom code and the VMEWindow product and subsequent development were poled in deriving this input. The percent of code thrown away due to requirements volatility was set at zero because the design requirements were based on the functionality of the old system and were confined to addressing the equivalent replacement of this functionality only and the capabilities inherent in the VMEWindow product that were not used

were not removed. In general, the development effort moved from a baseline of piece part functionality written in custom VxWorks to a cohesive commercial product that provided much of the required functionality. It was estimated that the custom code would require 75 percent design, 80 percent code modification and represent approximately 60 percent modification to the original integration effort to incorporate in order to meet the ground station requirements. Based on a yearlong evaluation of the custom code between the first and second software releases a subjective average weighting of "low" or 40 was derived for "Software Understanding" of the custom code. One would think that a custom code development would require zero percent Assessment and Assimilation to be used as part of the final application but this was not the case in our custom development. After the requirements analysis described earlier, it was discovered that some of the code provided functionality only meeting part of the requirement or provided the function in a manner that was not acceptable to the users. This resulted in a rating of 2 for Assessment and Assimilation. The USA programmers felt, after a year of evaluation, that they were considerably familiar (0.6) with the custom code. The adaptation of VMEWindow to the Space Shuttle telemetry ground station replacement or upgrade was rated by USA and APLabs software development team members. It was estimated that VMEWindow code would require 5 percent design, 30 percent code modification and represent approximately 10 percent modification to the original integration effort to incorporate in order to meet the ground station requirements. Based on developer training the team derived a subjective average weighting of "high" or 20 for "Software Understanding" of VMEWindows. The team considered peculiar Shuttle operational support requirements and additional documentation and testing requirements in deriving a rating for Assessment and Assimilation for VMEWindow. This resulted in a rating of 6 for Assessment and Assimilation. The team felt that a rating of completely unfamiliar was appropriate because the USA programmers had never seen the source code of the product prior to this development effort.

The model was run from 300KLOC to 900KLOC, first based on continued development of the custom code and second based on the actual baseline of VMEWindow. At all SLOC settings the model estimated a 76 percent reduction in software development effort.

The model was then run using the above input parameters to arrive at a composite estimated avoidance. Input parameters corresponding to the original state, both in skills and custom code baseline are compared to the combined skills and VMEWindows baseline. This resulted in a composite 83 percent estimated cost avoidance.

Conclusions

Comparing the actual project performance and the projected performance assuming continued development of the custom code reveals that cost and schedule were reduced. The technical evaluation of the product delivered revealed a marked improvement in every attribute within all three categories evaluated. Several additional benefits included technology transfer, continued collaborative enhancement of the product and the ability to address obsolescence.

Comparison of the estimated percentage cost avoidance based on empirical data and expert judgment corresponds closely to the composite percent estimated cost avoidance output from the COCOMO II software-estimating tool. The empirical analysis resulted in an estimate of 85 percent and the estimating tool resulted in an estimate of 83 percent. This correlation lends credence to the assertion that skill mix changes coupled with effective commercial product leveraging were responsible for the cost avoidance.

This methodology represents significant improvement over the status quo and should be evaluated for implementation on future and ongoing NASA and DOD software development projects. It is the

authors considered opinion the results warrant application of this methodology where close counterparts are available in industry. Strong close technical counterparts for data warehousing, recording, retrieval, command, control, data monitoring, network traffic generation and system administration functions can be found in industry. There are many NASA and DOD programs currently in work or soon to be out for bid that could adopt this type of development and sustaining engineering technique.

This study and pathfinder indicates that this would result in considerable savings to the Government and development of technologies that are directly applicable to the collaborator's competitive edge thus enhancing the return to the taxpayer. The ancillary missions of NASA and DOD to partner with industry for the benefit of industry while reducing costs and providing superior products to operations would seem to demand evaluation of this technique in any upgrade or new development project.

ACKNOWLEDGMENTS

The author gratefully acknowledges the following individuals and organizations: Randy L. Kirby, Stephen G. Prenger, Doug Johnson, Richard Weimer, Wayne Craig, Andrew Greenwood, Vada Maupin, Jeff Phefferkorn, Richard Price, Dennis Wesbecker, Patricia Hart, Bill Goodson, Jonathan Morsics, Tom Perez and Charles H. Fricker of United Space Alliance and Son Quach, Andy Mason, Bill Raney and Paul Lechese of AP Labs, the Integrated Data Systems Management Team and the Shuttle Telemetry Ground Station Upgrade development team.

This work was performed under the auspices of the Space Flight Operations Contract (SFOC, NAS 9-2000).