Statistical Process Control Meets Earned Value

by Walt Lipke and Jeff Vaughn
Oklahoma City Air Logistics Center

Levels 4 and 5 of the Software Engineering Institute Software Capability Maturity Model (SEI CMM®) imply the application of Statistical Process Control (SPC) to software management. SEI staff members have published a book [1], and are teaching a course [2] on the subject. Several software organizations are trying to apply SPC to quality control. This article expands the area of application. It presents an approach for software production management (i.e., cost and schedule control.)

The Test Program Set and Industrial Automation Branches of the Oklahoma City Air Logistics Center, Directorate of Aircraft Management, Software Division achieved Level 4 of the SEI CMM®)on November 7, 1996. At that time, and continuing today, this software group applies several measures in the control of its process and product output. The measures relate to financial health, project management, workload and labor, and process improvement (rework and productivity). Rudimentary SPC has been applied to the measures for some time in the form of run charts [1 and 3] (i.e., charts that graphically portray measured results in chronological sequence).

Run charts are fine, they provide trend information. But in the traditional quality application, what is expected are control charts [1 and 3]. From classes [2 and 4], conference presentations [5], and books [1 and 3], it is implied that *you are not really applying SPC unless you are using control charts*. Here is where *six sigma* originates. The predominant thought is you cannot have *six sigma* (translation: *really good*) quality unless you know the process is in control. Understanding whether or not the process is in control comes from the use of control charts, thus, the impetus to apply this SPC technique to software quality control. It is our understanding that the software organizations attempting to apply SPC are using data taken from product reviews, predominantly directed to the coding portion of the process. They are using defects identified in relation to effort expended, or lines of code or function points as data for the control charts.

The application of SPC Control Charts in this article does not focus on analysis of software defects. The following discussion will illustrate how SPC can be coupled to Earned Value indicators to provide information about the quality of project performance. The use of SPC Control Charts then becomes a tool in the control of software project cost and schedule.

Earned Value

The indicators from Earned Value (EV) Management, which directly relate to efficiency of project execution, are the Cost Performance Index (CPI) and the Schedule Performance Index (SPI). Their definitions are:

CPI = BCWP/ACWP (where BCWP is the budgeted cost of work performed, and ACWP is the actual cost of work performed).

SPI = BCWP / BCWS

(where BCWS is the budgeted cost of work scheduled).

For additional information and explanation concerning these formulas and terms, please refer to [6].

These two indicators, taken together, can be used to manage project performance [7]. They can provide very insightful information for managers regarding the status of their project. As described in the referenced article [7], when the inverse values of CPI and SPI (CPI^{-1} and SPI^{-1}) are compared to their respective cost and schedule ratios and the results are paired, one of nine recommended management actions is determined (see Table 1). As discussed in the article [7], the management actions are related to four possible strategies:

- Adjusting overtime or number of employees.
- Realigning employees to increase efficiency.
- Reducing performance requirements.
- Negotiating additional funding or schedule.

With this background, the Earned Value indicators, CPI^{-1} and SPI^{-1}, were chosen for SPC application. One very good feature of these EV indicators is they are *normalized*. Regardless of software project conditions (e.g., size of project, experience of staff, software engineering environment, programming language, etc), their ideal value is 1.0. Because they are *normalized*, many of the issues with applying SPC to software, such as variability and homogeneity of the data, are avoided.

Statistical Process Control

Software project managers normally assess their project status on some periodic basis. In our organization, we perform project reviews monthly. The project data for CPI and SPI is aggregated from the individual developers, then computed, charted and analyzed monthly along with several other indicators. Because the set of project data for analysis of each indicator (CPI^{-1} and SPI^{-1}) has only one data point per month, the type of SPC Control Chart selected is XmR, or individuals and moving range [1 and 3].

Table 1. *Management Actions*

CR vs. CPI^{-1}	SR vs. SPI^{-1}	Management Actions
Green	Green	Reward Employees
Green	Yellow	Increase OT
Green	Red	Increase OT or People
Yellow	Green	Decrease OT
Yellow	Yellow	Review & Adjust Assignments
Yellow	Red	Adjust Assignments; Consider Negotiation (Schedule)
Red	Green	Decrease OT or People
Red	Yellow	Adjust Assignments; Consider Negotiation (Funding)
Red	Red	Negotiation (Funding/Schedule/Rqmts); Fire Manager

For this control charting method, the individual values of monthly project performance are plotted in their sequential order. The average of all the values is calculated and, likewise, shown. upper and lower natural process limits (UNPL and LNPL) are also shown as distinctive lines on the chart. These lines are computed to be the *six sigma* limits of the process under review. Statistical theory provides methods to calculate the UNPL and LNPL based upon the dispersion of the moving range (mR) [1 and 3]. For the application of SPC presented here, the differences between the successive monthly values for CPI^{-1} and SPI^{-1} become the data for the mR analysis.

Just as for X, the moving range is graphed. Data points are plotted in proper monthly sequence, with the computed average value of mR. As with UNPL and LNPL lines shown on the Individuals chart, the UCL and LCL are displayed as lines on the mR chart. As for UNPL and LNPL, statistical theory provides computational means for determining UCL and LCL values.

The formulas for calculating the process, or control, limits (the *six sigma* values) of the XmR charts are available in the cited text references. In our application, because the adjacent X data points are paired to form the mR data, the subgroup size (n) is said to be two. Knowing n=2, the values of the constants required by the formulas are determined from the control chart tables [1 and 3]: $d_2 = 1.128$, $D_3 = 0$, $D_4 = 3.268$.

An example of the XmR chart is illustrated by Figure 1. Note, on the Individuals chart there is another line in addition to those for the average value of X, UNPL and LNPL. This line is labeled USL (i.e., the Upper Specification Limit). The USL is not derivable from the data; it is a performance value, or constraint, that the process is not to exceed. A considerable amount of subsequent discussion concerns USL and its value in relation to "X-bar," or the average value of X, and the UNPL.

Analysis/Interpretation

The successful project manager must continually ask, "Can the project be completed if it continues performing as it has?" SPC application to CPI^{-1} and SPI^{-1} can help answer the question. Comparing the X-bar values of these EV indicators to the planned performance (i.e., to the value of 1.0) provides information about current and future performance for the project. If the value is 1.0 or less, then the project is performing well and can be expected to complete within its planned cost and schedule. If it is greater than 1.0, then there may be trouble requiring management attention.

Now we are ready to discuss the upper specification limit, or the process constraint, mentioned earlier. The USL for cost is the *cost ratio* (CR), while for schedule it is the *schedule ratio* (SR). The *cost ratio* is defined as the total funding available for the project divided by the planned value (in EV terminology, budget at completion, or BAC). The *schedule ratio* definition is the negotiated period of performance divided by the planned period of performance. The value of both ratios may exceed 1.0; the portion of the ratio in excess of 1.0 establishes the amount of management reserve available for handling project risks [7].

Comparing the appropriate X-bar value to its respective USL (i.e., CPI^{-1} to CR, and SPI^{-1} to SR) provides information to the project manager as to whether or not the project is executable if

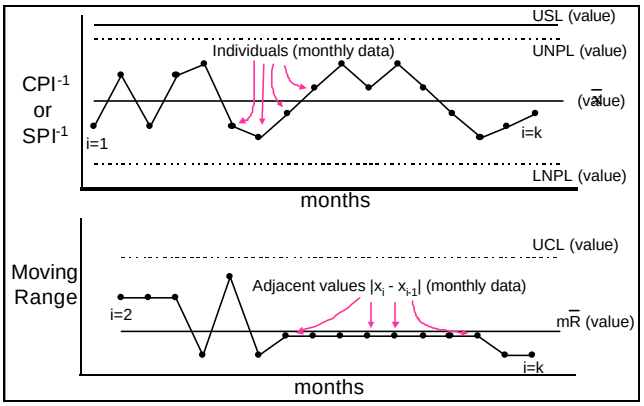


Figure 1. *XmR Example*

present performance continues (see Figure 2). This comparison is akin to the SPC analysis of process capability. In the manufacturing application, if UNPL and LNPL are within the USL and LSL, the process is said to be *capable*. If the USL, or LSL, are within the UNPL, or LNPL, calculations can be made to determine the probability of producing defective products. Corrections to the process are sought to minimize the output of defectives.

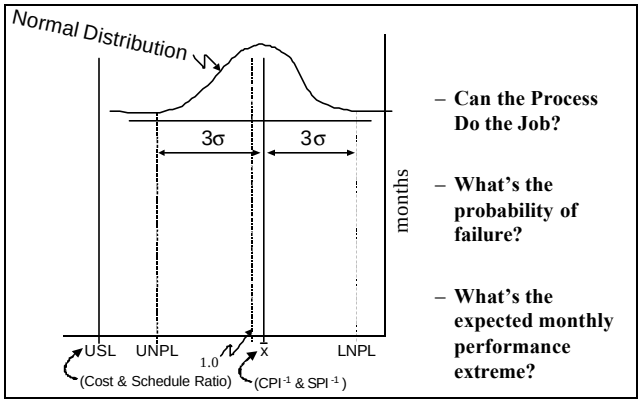
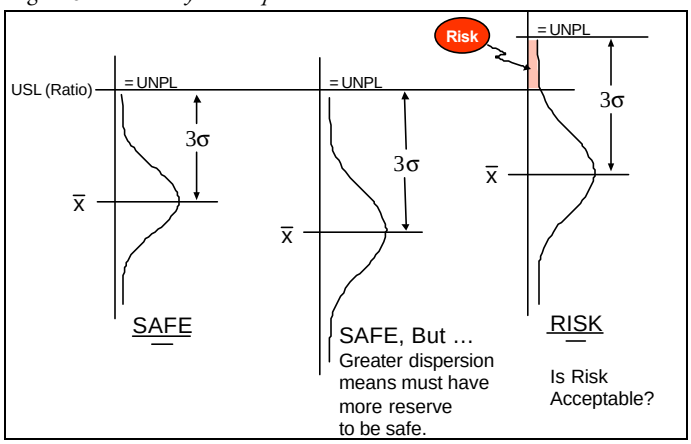


Figure 2. *Process Capability*

The interpretation of the SPC application of process capability to software project management, and the EV indicators CPI^{-1} and SPI^{-1}, is somewhat different from the description given for manufacturing (see Figure 3). The project can have *defective* monthly performance results and still be in good shape. The process can be expected to achieve satisfactory results if the average value of CPI^{-1}, or SPI^{-1}, is less than the USL (cost or schedule ratio). Besides determining process capability, there is

Figure 3. *SPC Analysis/Interpretation*



other interesting and useful information from the SPC analysis; the probability of project failure can be determined, along with the expected monthly performance extreme. The UNPL value is the expected performance extreme. The portion of the *normal distribution* that exceeds the USL quantifies the risk of failure.

A few other observations concerning Figure 3 can be made. If UNPL equals USL, the project performance could be termed *safe* (i.e., the risk of failure is nil). If 3 sigma is large with respect to X-bar, then there is a large amount of dispersion (mR) in the monthly performance. To be *safe* with large dispersion requires a greater amount of management reserve. To be competitive, it is very advantageous to minimize the UNPL and move it towards perfection (i.e., the value 1.0). The risk, or probability, of non-performance should be minimized. By decreasing risk and, thus, management reserve, a company can decrease its bid price, and increase its chance of contract award. Also, risk can be planned for a project. The project can be managed to that amount of risk. Often that is what is done to win the bid. But, without using SPC, the bidder does not know his chance of failing. *If the bidder plans no Management Reserve, his probability of achieving the project plan is only 50 percent.* This point is easily seen from Figure 3 by lowering USL until it equals the planned performance of CPI^{-1} or SPI^{-1} (i.e. 1.0). Even if the developer has a very good process and the process variability is minimal, without planned reserve, his chance of achieving cost and schedule is only 50 percent. Fifty percent is not very good odds if the company wants to stay in business and make money.

Project Manager Use

Earlier, we alluded to the application of SPI and CPI in building a project plan. Past statistical performance can be used to build a risk strategy leading to the requirement for management reserve. The following discussion, however, will focus on project performance instead. Project performance evaluation is simple and is very similar to the description given in reference [7]. The evaluation for the SPC application to the inverse of CPI and SPI is a comparison of the average values to the planned value (1.0) and the USL (cost and schedule ratios). The evaluation criteria are shown in Table 2. If the X-bar value is less than or equal to 1.0, the project can be expected to complete as planned. If the value is greater than 1.0, but less than or equal to the USL, then the project can be expected to complete within its allocated cost and schedule. Of course, in this range of performance some of the management reserve is being consumed by execution inefficiency. Finally, if the X-bar value is greater than the USL, project failure can be expected. For ease of recognition, the status can be color coded *green*, *yellow*, or *red*. *Green* indicates the project can complete within the plan, while *yellow* means the project can be completed within the cost and schedule allocations (i.e., within the plan plus management reserve), and *red* says failure is to be expected.

Table 2. *Evaluation criteria for X-bar cost and X-bar schedule*

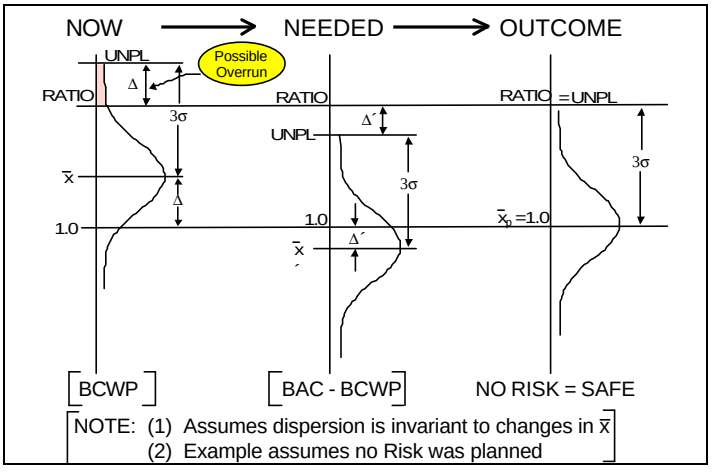
$\bar{x} \leq 1$	GREEN	Project can be completed as planned
$1 < \bar{x} \leq USL$	YELLOW	Project Manager and employees get to keep their jobs
$\bar{x} > USL$	RED	A bad situation for those involved

Once the color status has been determined for cost and schedule performance, Table 1 can be used to determine needed management action. Getting to corrective action in Table 1 is a fairly simple matter. *Red* performance status requires management attention. If the project is not performing well, something must be done; corrective action is needed. Eyebrows will raise when *yellow* occurs, but a more in-depth look should be made before any correction is made. Evaluating the performance of CPI^{-1} and SPI^{-1} using statistical methods leads to appropriate actions.

Beyond understanding what to do to bring the project back in line, software project managers need to know the extent of correction necessary. Specific areas in which SPC can be used to quantify correction are adjustments to overtime and staffing, and funding and schedule negotiations. If the project manager has resigned himself to negotiation for correcting the project's ills, the amount of overrun of funding or schedule with respect to the customer agreement can be easily determined. Overrun is found by simply multiplying the difference between the X-bar value and its respective USL times the BAC for funding, or planned period of performance for schedule. The quantity calculated is the amount needed to raise the probability of meeting these revised performance requirements to 50 percent, thus, the minimum amount to be pursued in the negotiation. Additional funding or schedule must be added to the minimum value if the desire is to provide some amount of management reserve for the remaining portion of the project. In general, these negotiations are not easy. The developer has a tendency to understate real needs of the project. By not coming to terms with actual performance, he settles for less than the full amount needed to successfully deliver the product, and ends up having to negotiate again. Generally, the second negotiation is considerably more unpleasant than the first. Unless there is no other source for the products or services, the software organization will not likely be awarded another contract by the customer.

The corrective action concept for adjusting overtime and staffing is illustrated by Figure 4. If efficiency has been poor, something must be done for the remainder of the project to raise

Figure 4. *Project Manager Application*



the efficiency so the project completes within allocated cost and schedule. The possible overrun is symbolized in the figure by Δ (delta), and the amount of correction needed for the remainder of the project is symbolized by Δ' (delta/prime). The Δ condition has existed over a portion of the project (i.e., BCWP, the earned value for the tasks completed or in work). The Δ' condition will need to occur over the project remainder (BAC – BCWP) to bring the overall performance to the desired state. Figure 4 illustrates the theory; however, it implies that poor performance can be corrected to the ideal performance condition. Please disregard the implication. *Corrective action is not free*. It is achieved at the expense of cost or schedule. You may be able to correct schedule, but it will be at the increase of cost. Unless true efficiency gains are achieved, management reserve is expended for the corrections.

The method for calculating the X-bar value needed for achieving project cost and schedule requirements is simple, and based upon the same concept as the *To Complete* indices used in EV Management [6]. The calculation method is depicted in Table 3. The result of the correction X-bar (', X-bar/prime) is used in the overtime and staffing equations shown in Table 4. The strategy for recovery can be banded by recalculating X-bar/prime, using total funding available in place of BAC. Recalculating the overtime and staffing adjustments with the extreme X-bar/prime value determines the minimum adjustments the project can make and still achieve its negotiated cost and schedule.

Project Changes

Legitimate questions regarding this application of SPC are "What happens when the project is replanned?," and "Can the data from the XmR chart prior to the replan be used with the data obtained afterwards?" Obviously, the answers come from the changes caused by the replan to the software project's EV system. If (think of this as a manufacturing example) the only change is in the quantity of products, it follows, there will be no adjustments made to the work breakdown structure (WBS) or the earned value of the tasks. For this type of replan, the old data can be used with the new. If, however, the WBS or the task values are changed by the replan, then the remainder of the project must be treated as though it is a new start.

Table 3. *Performance Correction Index*

- For Schedule or Cost "curve shifting":

$$\Delta = \bar{x} - 1.0 \qquad \text{[shift away from plan]}$$

$$\Delta' = \Delta \bullet \frac{BCWP}{BAC-BCWP} \qquad \text{[performance correction to achieve plan]}$$

$$\bar{x}' = 1.0 - \Delta' \qquad \text{[required performance index for remainder of project]}$$

Table 4. *Adjusting Overtime and Staffing*

- Schedule Recovery (Reserve Funding is possibly used)

$E_{SR} = (1 / \bar{x}'_{SCHED}) \bullet E_P$
where E_P = planned number of employees

$OT_{SR} = (1 / \bar{x}'_{SCHED}) \bullet (1 + OT_P) - 1$
where OT_P = planned overtime rate

- Cost Recovery (Schedule Reserve is used)

$E_{CR} = (\bar{x}'_{COST}) \bullet E_P$

$OT_{CR} = (\bar{x}'_{COST}) \bullet (1 + OT_P) - 1$

- Band the Recovery Strategy
  - Substitute Total Funding Available for BAC in the Δ' calculation

Application

In our organization, projects have been managed using the cumulative values of SPI and CPI for some time. Actual project monthly data is plentiful and readily available to create and test the SPC application described. One project's data is exhibited in Figures 5 and 6, the XmR charts for CPI^{-1} and SPI^{-1}, respectively. Although not shown, the histograms prepared from the performance results for both CPI^{-1} and SPI^{-1} approximate *normal* distributions. Although there are SPC experts who will argue that not having a normal distribution of the data does not invalidate the use of Control Charts, some confidence is created in this application when it is seen that the distributions appear *normal*.

To begin the SPC analysis, refer first to Figure 5 for the discussion of the CPI^{-1} Control Charts. As can be seen, the process can be said to be reasonably well controlled; the average value of mR is fairly small (0.2652). Even so, the variance is large enough to place the computed value of UNPL above the *cost ratio* (USL). Recalling the earlier analysis/interpretation discussion, when UNPL is greater than USL, there is a computable probability that

Figure 5. *Software Development Project CPI^{-1} Data*

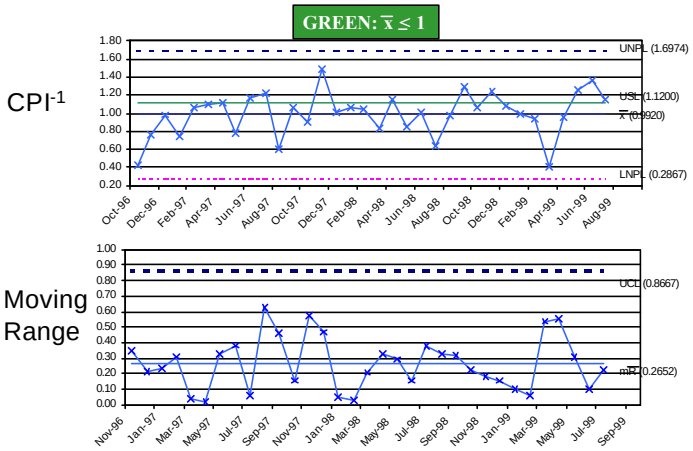
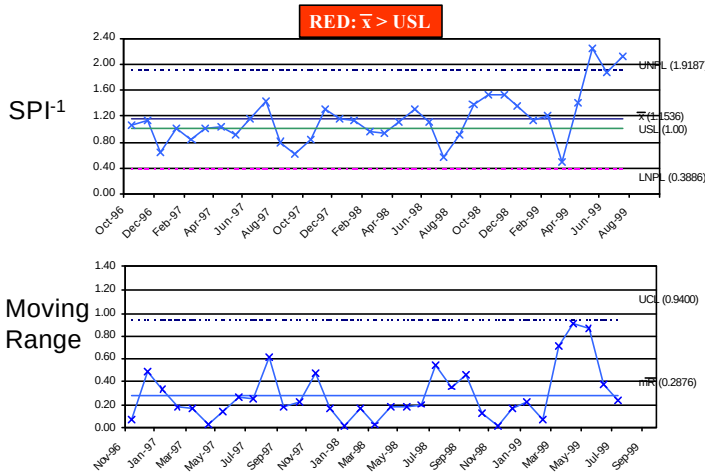


Figure 6. *Software Development Project SPI^{-1} Data*



the project will exceed its allocated cost. Performing the mathematics, and using the normal distribution table [8], the probability of overrunning the *cost ratio* is determined to be 29.3 percent. Observed directly from the CPI^{-1} graph, worst expected monthly performance (UNPL) is 1.694. The average value of CPI^{-1} (X-bar) is seen to be less than the *cost ratio*, so the process can be said to be *capable*. Likewise, X-bar is less than 1.0 and, thus, the status indicator color is *green*. The project manager can expect the project to be completed within its planned cost. With very good numbers, the project manager does not have much worry here.

Let us shift our attention to Figure 6. The control of schedule is not as good as for cost. The average value of mR (0.2876) is somewhat larger, and, correspondingly, the value for UNPL (1.9187) is computed to be larger than for the CPI^{-1} Control Charts. The average value of SPI^{-1} is greater than one, and greater than USL, a *red* status condition. Here is something to worry about—not meeting the customer's schedule. Using the *schedule ratio* and the average value of SPI^{-1}, and the normal distribution table [8], the probability of failure is found to be 72.7 percent. This is definitely not good; the condition needs management action. Before the recommended action is described, a few observations can be made. *This project planned no management reserve for schedule (USL=1.0).* The only way this project can correct its poor schedule performance, other than having a miraculous improvement in schedule efficiency, is through the use of the cost reserve. It is always better to have reserve in both cost and schedule.

Returning to Figure 6, focus on the last three data points (May, June, and July '99) of the SPI^{-1} graph. They are especially interesting. These points, which hover around UNPL, are anomalous. The probability of these data points occurring is infinitesimal. Their existence is virtually impossible. From SPC theory, this behavior leads the analysis to seek an assignable cause [1 and 3], some influence outside of the system. Reviewing the same months on the CPI^{-1} graph, the data appears to be reasonable. From EV analysis, we know, if CPI performance is good and SPI is poor, then the project is likely suffering from a manpower shortage. Actually, the project had several software engineers hired away (by a single employer) in those months and, correspondingly, the manager was unable to hire a sufficient number of replacement employees. Also, we found there were other significant contributors to the poor schedule performance: the impact of the huge Oklahoma City tornado that affected many of our people from May through July; unplanned Air Force-driven training; and, a computer security alert that required the staff's attention. In proceeding with the analysis, a decision should be made as to whether to include the data attributable to an assignable cause. Because our purpose is merely to demonstrate the calculations, we have chosen to retain the data.

From Table 1, management's correction strategy is *increase overtime or staffing*. Next, the schedule recovery formulas of Table 4 are used to quantify the necessary increases. The formulas require the value of the schedule correction index, X-bar /prime. Knowing the values for X-bar (1.1536), and BCWP/BAC (.617), the completed portion of the project, X-bar/prime is computed to be 0.7526 from the equation given in Table 3. Having the value for X-bar/prime, we can determine if the schedule correction can be accomplished by simply increasing overtime. For the completed portion of the project, the effective overtime rate is 7 percent. The rate required for the remainder of the project is determined to be nearly 42 percent. Working employees at 42 percent overtime for very long is not advisable. Doing so will likely increase the loss of employees and worsen the problem. More employees are needed. Dividing the effective number (59) of employees for the completed portion of the project by X-bar/prime (0.7526) yields 78, the number of employees needed for the remainder of the project. It is interesting to note that the project manager, in his effort to determine the corrective action, reported the same number of employees after spending two days manipulating a commercial project scheduler. Certainly, this single correspondence with a commercial project scheduler is not enough evidence to say the methods described here will always provide good management information, but it does greatly increase the level of confidence. Also, the 10 minutes we spent making the estimate compares very favorably to the two days needed by the project manager to come to the same result.

Summary

The concepts of statistical process control, specifically control charts, have been discussed with respect to software project management of cost and schedule. SPC Control Charts are shown to have a practical application to the EV indicators, Cost and Schedule Performance Indices. SPC can be easily utilized by software projects using Earned Value Management. Furthermore, example results discussed indicate this application of SPC can be useful. It appears that coupling SPC Control Charts to EV has the potential to be extremely powerful. The methods and techniques described can be used for:

- Quantifying Problems.
- Process Capability.
- Probability of Failure.
- Worst Expected Monthly Performance.
- Developing Recovery Strategies.
- Overtime.
- Staffing.
- Negotiation Values.
- Project Planning.
- Quantifying Risk.

Final Thoughts

Beyond project planning and control application described in this article, implicit in SPC is *process improvement.* An example of *assignable cause* was provided in the discussion, but just as important is the improvement of *common cause* entities [1, 2]. Improvement to common causes reduces variability of the process and, thus, risk. For the software development organization, the reduction of risk leads to decreasing the requirement for management reserve, thereby making the organization more competitive. And for the customer, reduced variability in the software developer's process means lower prices and greater probability of achieving cost and schedule. Everyone wins—it is powerful stuff.

▸▸ *This article is continued on page 28.* ▸▸

portions. In the above Kalman gain equation, for example, intermediate expressions for PH^T are computed first, followed by $[HPH^T + R]$. Then $[HPH^T + R]^{-1}$ is derived and that result premultiplied by the PH^T term that was already expanded. Although this requires somewhat more work due to the maintaining of intermediate results, overall savings and code generation metrics are still very impressive. Depending on the particular implementation and the correlation of the measurements, FLOP savings on the order of 4:1 to 6:1 and more have been tabulated. These savings were achieved with significantly less effort and higher reliability than the manually tedious and error-prone traditional approach.

A process similar to the above is then used to unit test the algorithm in the target language. A test driver is created that executes the unit under test. Its executable image is called in place of the expanded code bracketed in the example m-file above. The MATLAB script and the target language test driver must both provide for the reading and writing of data files to effect the transfers of test data and results. The author has elected to mechanize these transfers in IEEE 64-bit binary format to take full advantage of the numeric capabilities of MATLAB. An outline of the MATLAB script was shown in Figure 2.

Although applications vary enormously in complexity, the author has experienced estimated savings from 40 to almost 80 percent in development and testing time (including debugging effort). For example, it took approximately 20 hours to manually develop and test code to decrement a 6-state covariance matrix and about 16 hours for the 6-state Kalman gain matrix. Applying the techniques outlined in this paper reduced these efforts to approximately 12 and eight hours, respectively. Without manually deriving corresponding 9-state equations, a fairly daunting task, extrapolations from actual 6-state manual results and other similar experience were used to estimate the level of effort that would be required for the manual derivation of 9-state equations. The following table summarizes and compares both approaches.

	Pencil & Paper	MathCAD & MATLAB	Estimated Savings
Decrement Covariance $[I-KH]P[I-KH]^T + KRK^T$			
6-State	20	12	40.0%
9-State	56*	16	71.4%
Compute Kalman Gain $PH^T[HPH^T + R]^{-1}$			
6-State	16	8	50.0%
9-State	42*	10	76.2%

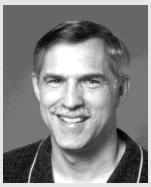
* Extrapolated from similar 6-State experience

Table 1. *Estimated Effort to Code and Test (hours)*

Summary

The use of commercial tools, such as MATLAB and MathCAD, can dramatically improve the efficiency of the software development process as well as the reliability of the final embedded product. Since systems engineering groups are increasingly using these tools to develop and document their products, it is becoming imperative that software engineers acquire the necessary proficiency to use and integrate these products into their processes. This is especially critical as contractors strive to remain competitive while meeting exigent schedules and maintaining budgetary constraints. Although not a panacea for all that is ailing in the software development process, the logical application of available commercial tools can have a significant, positive impact on that effort. This paper outlines such a process that the author has successfully applied on a current development program.

About the Author



Keith R. Wegner is a Fellow Engineer in the software engineering group at Northrup Grumman Corporation's Electronic Sensors and Systems Sector in Baltimore. He has a master's degree in electrical engineering from the Johns Hopkins University with emphasis in signal processing and control systems. He has used MATLAB for approximately 14 years.

Northrup Grumman Corp.
P.O. Box 746, MS-429
Baltimore, Md. 21203
Voice: 410-765-4664
Fax: 410-765-1492
E-mail: keith_r_wegner@mail.northgrum.com

▸▸Statistical Process Control Meets Earned Value, by Lipke/Vaughn, continued from page 20▸▸

References

1. Florac, William A., and Anita D. Carleton, *Measuring the Software Process*, Addison Wesley, Reading, Mass., 1999.
2. Pitt, Hy, *SPC for the Rest of Us*, Addison-Wesley, Reading, Mass., 1995.
3. Software Engineering Institute Course, Statistical Process Control for Software, July 1999.
4. Software Productivity Consortium Course, Statistical Process Control and Quality Management Techniques, August 1998.
5. Radice, Ron, Statistical Process Control for Software Projects, 3rd Annual Software Metrics Conference, December 1997.
6. Fleming, Quentin W., *Cost/Schedule Control Systems Criteria, The Management Guide to C/SCSC,* Probus, Chicago, 1988.
7. Lipke, Walter H., Applying Management Reserve to Software Project Management, CrossTalk, March 1999.
8. Crow, Edwin L., Davis, Francis A., Maxfield, Margaret W., *Statistics Manual*, Dover Publications, New York, 1960.

Notes

1. To remove any confusion, by monthly performance values, we mean the values include only the performance occurring during the month.
2. The application of Table 1 in [7] required CPI^{-1} and SPI^{-1} to be cumulative values. For this application, CPI^{-1} and SPI^{-1} are average values of the monthly data.
3. The assumption in overtime and staffing equations is that the plan is being executed; i.e., the overtime rate and the staffing employed is in agreement with the project plan. If the effective values for either differ from the plan, it is recommended to use those values in the equations.
4. See the Table 1 management action description for the condition: cost comparison green, schedule comparison red.

About the Authors

Walt Lipke is the deputy chief of the Software Division at the Oklahoma City Air Logistics Center, which employs approximately 600 people, most of whom are electronics engineers. He has 30 years of experience in the development, maintenance, and management of software for automated testing of avionics. In 1993, with his guidance, the Test Program Set and Industrial Automation (TPS and IA) functions of the division became the first Air Force activity to achieve Software Engineering Institute Capability Maturity Model (SEI CMM®) Level 2. In 1996, these functions became the first software