Metrics That Matter in Software Integration Testing Labs

Christian Hagen, A.T. Kearney
Steven Hurt, A.T. Kearney
Andrew Williams, A.T. Kearney

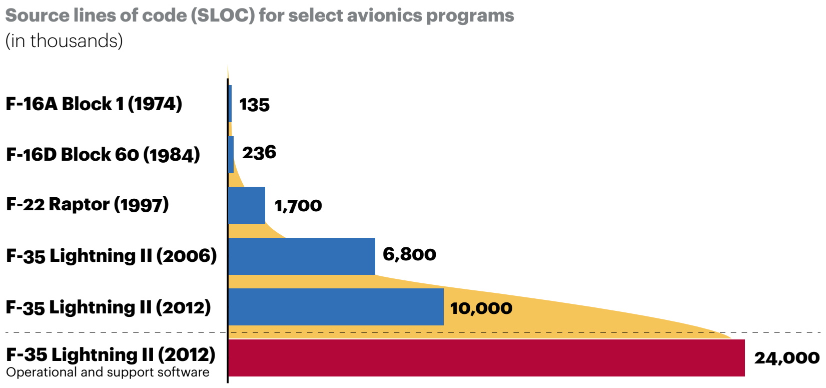
Abstract. Without having in place data-driven metrics that give a holistic business perspective of software integration testing laboratories, leaders of the DoD's weapons programs are unable to optimize the performance of these full-system and subsystem integration labs that test and certify integrated hardware and software during the development, modernization, and sustainment of the U.S. military's integrated and complex systems. Yet these metrics are not available across the DoD lab footprint, even though the labs' significance is growing in parallel with the military's rapid shift from using equipment with capabilities based on advanced hardware to equipment that is dependent on fully integrated, complex systems of both hardware and software.

This game-changing shift is now evident in weapons programs throughout the military. It's seen, for example, in the F-35 program, whose software over the years has increased in size to approximately 24 million source lines of code, which has made the testing more difficult and led, in part, to the program's multiyear delays (see Figure 1).

As this shift intensifies, so too does the program leaders' need to successfully compare the operations of any lab to that of any other lab, each of which today approaches software integration testing with its own particular processes for measuring its progress and success. They simply need to be able to answer several pertinent questions that they cannot answer now, such as the following:

- Is the lab running at the appropriate level of efficiency, effectiveness, and cost, and if not, how can the current level of performance be improved?
- Can the lab handle additional testing and, if so, how much should be moved there and from where?
- Should the lab be updated and, if so, how much money should be spent on the update?

Figure 1
The amount of software in military avionics systems has skyrocketed



Note: SLOC for F-16 and F-22 are at first operational flight. F-35 SLOC figures are from first test flight and current estimates/sources.
Source: Hagen, C., Hurt, S., Sorenson, J. "Effective Approaches for Delivering Affordable Military Software." CrossTalk – The Journal of Defense Software Engineering, Vol. 26 No. 6 (November-December 2013).

- Should the lab be closed rather than updated and should the testing be transferred to another facility?
- Would the labs see reduced costs and improved performance with the purchase of new, faster testing equipment?

The Need for Metrics That Matter

The need to have metrics that matter when making decisions facing software integration labs was recently underscored by Robert Ferguson of Carnegie Mellon University's Software Engineering Institute (SEI). Ferguson's 2012 research shows that using project dashboards and following the right measurements are critical to project management because they provide managers with the information they need to perform different tasks at the correct times—much like the dashboard of a car leads to a successful journey by preventing the driver's running out of gas, going over the speed limit, or arriving late [1].

Ferguson showed a project dashboard—which should be constructed to suggest different decisions about product quality, and about directing and controlling the work—provides measures for the critical areas of project decision making. These measures include scheduling, resource allocation, scope and change, product quality, and effective process performance. In addition, the dashboard should do the following:

- Forecast milestones and delivery of scope
- Provide clear warnings if the plan is not working or an unplanned event has affected some desired outcome
- Support re-estimation and re-planning by showing the magnitude of the problem

Measures like these have never been more important than in today's environment in which software integration labs deliver the software and hardware capability weapon programs must have to win on the battlefield—a critical role the labs can perform only if they have the metrics they need.

Unfortunately, most software and program leaders today are attempting to make decisions without metrics that matter to them. Primarily, they have engineering- and technology-based metrics—metrics that are of value to those who care mostly about being able to test a single piece of equipment, not manage the overall operation of a lab or group of labs. What they need are metrics that are valid to those who must make command-level decisions from a holistic business perspective.

When DoD leaders try to make decisions like moving one lab's testing to other labs, they run headlong into major problems caused by the lack of metrics that really matter. Since each lab measures its progress and success with its own unique processes, decisions across the footprint are made using a nonstandard, and often ad hoc, approach. And with no standard set of metrics, leaders are uncertain about what the available metrics mean, which ones matter, and how they can use them to make fully informed command decisions about the system integration labs. Their confusion is compounded by the lab contractors' belief that since the labs use different technologies, test different equipment, and have completely different workloads, they cannot provide the metrics needed to compare operations—a belief that has been proven groundless in many other industries.

It was this confusion that led the leaders of a major avionics program in the DoD to determine they needed to significantly improve the way they looked across multiple labs to compare operating costs, performance, and other key metrics. Through research they concluded that the metrics that matter the most for use in the system integration labs would come from examining operations with similar capital-intensive processes.

They found the metrics they needed in manufacturing and operations, which has long been using a standard set of metrics—capacity, efficiency, effectiveness, and capability—to compare the operations of manufacturing plants, regardless of what the plants were producing. While the type of work done in these manufacturing facilities—inputting parts, assembling them, and outputting completed products—differs greatly from that of the integration labs—inputting software code and hardware, running tests against the code, and putting out a report on whether the code is good or bad—the processes are similar. Therefore, the metrics can be similar as well (see Figure 2). Although much has been written about software estimation and quality, [2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21], much less has been written about software integration testing.

The metrics found in these plants are derived from the body of work in manufacturing excellence that crosses many industries having similar processes, although a variety of products. They also are perfectly applicable to software integration labs. And now a few decision makers across the labs are starting to discover that these metrics—capacity, efficiency, effectiveness, and capability—enable them to not only measure and improve each software lab's cost and performance but to effectively manage all their labs as they test software systems that are fast becoming the strategic weapons on which the military's future success depends.

These four metrics closely resemble those of the overall equipment effectiveness (OEE) framework that was developed over the years to measure how effectively a process was executed in a manufacturing facility, and is now being used across industries, including the automotive sector. This framework was designed to give leaders the metrics to compare processes across factories and industries—the metrics they simply have to understand if they are to manage effectively their businesses and operations.

OEE, as research shows, is based on a standard set of metrics for understanding the manufacturing process [22]. It is captured through the following formula: OEE = machine availability x machine performance x product quality. The result is presented as a percentage that can be used to understand how the current manufacturing process of a plant is performing, and to determine how one or all three of these factors can be changed to improve this performance. It can also be used to compare performance across manufacturing plants within a company, throughout an industry, or across industries.

Additional research has shown that OEE can be applied more generally to operations, plants, and machinery. An article from 2003, for example, shows how cross-functional teams can apply OEE principles to multiple areas of operations and further shows that OEE principles can be applied to areas beyond manufacturing, pointing the way to its application in software integration labs [23].

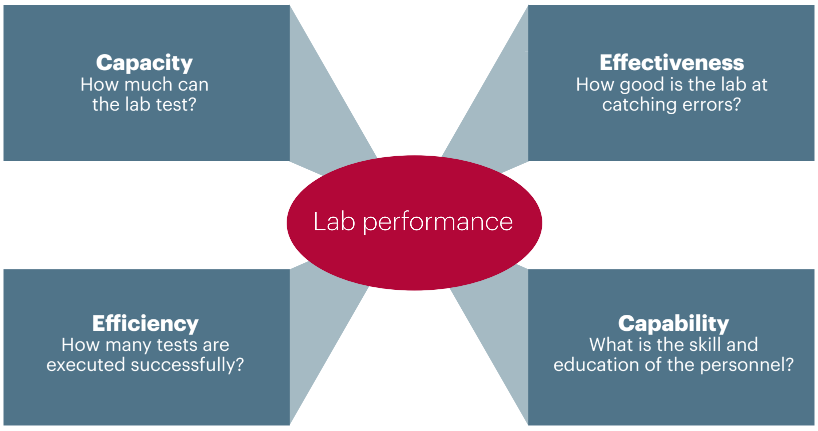
Leaders in the avionics program tweaked this framework for use within the department's software integration laboratories. They have learned that the four metrics—capacity, efficiency, effectiveness, and capability—are easily transferable to the labs because their measurements are directly comparable to those made in the automotive factories (see Figure 3).

Figure 2
The metrics for manufacturing and for software testing labs are similar

	Production metrics	
	Manufacturing	**Software integration lab**
Capacity	• Number of cars produced per hour	• Number of test points executed per hour
Efficiency	• Number of good cars produced per hour	• Number of tests executed on condition
Effectiveness	• Number of quality fixes	• Number of defects found
Capability	• What can the factory produce? (for example, Porsche vs. Yugo)	• What areas and complexity of tests can the lab execute?

Source: A.T. Kearney analysis

Figure 3
Software metrics for laboratory performance


Capacity
How much can the lab test?
Effectiveness
How good is the lab at catching errors?
Lab performance
Efficiency
How many tests are executed successfully?
Capability
What is the skill and education of the personnel?

Source: A.T. Kearney analysis

Capacity

While automotive factories count their output of vehicles per hour, software integration labs measure capacity by counting the number of tests executed per hour.

This metric, measured in test points, is the throughput per hour in terms of a lab's ability to execute its raw work. Test points, which at a basic level represent specific criteria to evaluate for validation and successful testing (for example, specific engineering performance values or, for a smoke test, the expected system output to a standard set of inputs), are used as a basis for the starting point for lab capacity. Test points are executed within a variety of test types, such as integration, verification, and regression tests. They are a measure of how much work, in total units, could be accomplished if the lab worked nonstop around the clock.

Capacity is measured in test points, which can easily be converted into derivative metrics like shift, daily, and yearly capacity. And it serves as the best proxy for lab size, showing whether the lab equates to a factory that is big or small. Knowing this capacity will, among other things, help DoD leaders determine whether the work they want to shift to another lab can be handled by that lab or not, vis-à-vis capability or capacity.

Because test points are the basic unit of lab production, comparing dollars per test point is the core indicator of a lab's cost. Using this comparison, decision makers can determine, for example, the cost of running a test or of finding a defect—such as a major defect that would cause the postponement of an unmanned vehicle's mission or a minor defect that might cause the malfunction of a truck's power steering.

Efficiency

While automotive factories check the number of "lemons" produced each hour, software integration labs measure efficiency by checking the number of tests executed hourly "on condition."

This quality metric indicates how well a lab is doing its work. If it can do 100 units of work each day, but only 50 units, on average, are correct, the labs' efficiency metric would be quite low.

"On-condition" is a test executed successfully—a determination based on the checklist and setup procedures handed down by the system engineers. Since the test is successful, it does not need to be performed again. Efficiency measures the percentage of tests executed correctly—not whether the software being tested passed or failed the test—and is calculated by dividing test points on condition by total test points attempted. "Off-condition" is a test that must be repeated because an error occurred in testing methods or setup. A false "on-condition" test is properly executed on condition, but it must be repeated because further analysis shows the test package was poorly designed.

Tightly linked, lab capacity and efficiency are often measured together to provide a clear understanding of their combined effect. With baselines derived from this combination, leaders can begin making command-level decisions about issues such as how a given action would change the lab's throughput, how a different action would affect the lab's cost per hour or cost per defect, and how some other action would impact the lab's efficiency or capacity.

Effectiveness

While automotive factories count the number of quality assurance fixes, software integration labs measure effectiveness by counting the number of software defects.

This metric points out how good a lab is at discovering errors. If, for example, its primary purpose is to find defects or certify code, the number of work units to defects could be a measure of effectiveness.

Effectiveness is measured by the number of test points executed per defect found, and it is calculated by defect found divided by test points attempted. This measurement of the lab's testing procedure shows how many tests must be run before the lab starts finding errors in the testing procedure. This metric is based on the assumption that labs have the capability to properly test the code. Testing capability means having the subject matter expertise, the appropriate number of personnel, and the right equipment to test the code. Since this metric measures the lab's ability to find defects in an existing code base, the resulting output of the metric is driven by the quality of the code being tested by the lab. Since quality of code can vary greatly across different development types and teams, this metric should not be used as an absolute value to compare labs across different development types. Instead, the trending of this metric within projects of similar size and scope can help a lab administrator track a lab's performance compared to historical testing efforts.

In a white paper on their research into software defects, SEI's Julie Cohen, Robert Ferguson, and William Hayes show that classifying defects appropriately and tracking them differently can increase lab effectiveness [24]. They suggest quantifying the priority of addressing these defects by assigning a Risk Priority Number (RPN) to each defect, a number that is calculated with "three distinct attributes of failure sources": severity, "a rating of the adverse impact of the failure"; occurrence, "how often the source of failure is encountered"; and detection, "how detectable the failure is when it occurs."

While not specifically addressing defects in software integration labs, the SEI authors underscore the need to view the defect data in the appropriate way with the appropriate metrics. This approach is essential to solving the broader issues DoD leaders face as they try to more effectively manage labs across the footprints.

Capability

While automotive factories explore the functionality of their equipment and what each factory can make, software integration labs measure capability by exploring the ability of each lab to meet the overall requirements.

This metric is the skill set of a lab's workforce and the functionality of its equipment. It is used to compare how well each lab can test specific areas of the software and is the function of three factors:

- Knowledge, which is assessed across product, functions, and technology, and is proven through work experience requiring expertise in the product, function, and technology areas
- Competency, which is assessed across current work behaviors and skills required to perform the work and proven by the existence of artifacts, such as current job descriptions and training, which are used to validate managers' and directors' scores for their teams and specific knowledge areas
- Capacity, which is measured by the availability and readiness of the lab's resources (human and infrastructure) to perform an activity

Because capability is also directly affected by a lab's equipment composition, this composition must be analyzed in any lab-to-lab comparison.

Capability plays a major role in leaders' overall management decisions because it has an implicit effect on the other three metrics that matter. Therefore, its impact on each of these metrics must be understood before making changes to the size, experience, or skill set of the workforce.

Approach to Applying the Metrics That Matter

When the leaders of the avionics weapons program began to evaluate the current strategy for software integration labs and to explore alternative models that might deliver better value, they quickly learned they needed metrics on which to base decisions—data-driven metrics that matter. In order to complete the evaluation, the team developed metrics that illuminated each lab's actual performance. These metrics drove the assessment of software integration labs and enabled leaders to accurately measure and compare lab performance across the footprint. They made possible the direct lab comparisons for analysis and enabled the leaders to create a business case to model future-state scenarios and compare cost savings, transition risks, and steady-state capacity risks across scenarios (see Figure 4).

As an example of the power that using the correct metrics gives to business leaders evaluating testing labs, Figure 5 describes a hypothetical model of additional hours of test time required during a lab transition scenario. By understanding a lab's normal throughput in tests and operational efficiency (first time right execution), the effects of the overall program test hours can be estimated as different areas of the lab are shut down to transition.

In this example, the lab is transitioning in two phases, each of which will reduce the testing capacity of the lab during the transition time. Using the appropriate metrics the leaders can estimate the impact to the program and additional hours required to keep the same level of testing results as before the transition started (revised test hours).

Besides evaluating the labs' current strategy and exploring alternative value models, the assessment's specific objective was to reduce the labs' life cycle costs by moving the program's testing from its current location to potential alternatives and to do so without degrading current performance. The program also set out to answer questions about the attributes of the current lab footprint; about alternatives to the current lab environment; about the costs, benefits, and risks of the current plan and the proposed alternatives; and about the recommended strategy (current plan versus proposed alternatives).

The program met its objective with a thorough analytical review of the current long-term strategy and potential alternatives. In doing so, it determined that the best value alternative would result in the lowest life cycle cost with manageable risk while not degrading lab capabilities or performance.

Results

The metrics developed during the assessment provided the information needed for the leaders to recommend that the avionics program transition the testing to the alternative labs but maintain the current lab's performance and its operator and equipment capability. This result provided less risk during the transition as well as steady state. It also saved more than 30 percent in life cycle costs, for a total net present value savings of hundreds of millions of dollars (see Figure 6).

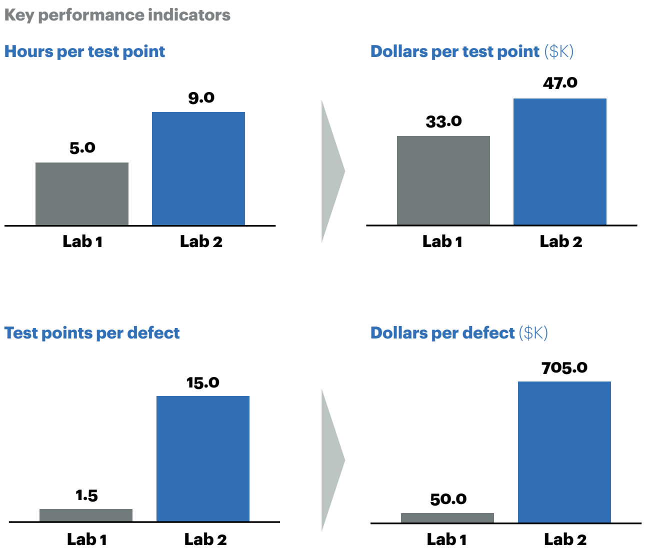
Using the metrics, the team modeled several courses of action through the perceived end-of-life. From these, it recommended a clear course of action for moving the testing, including the expected cost savings, transition risks, and potential risks.

The clear, communicable metrics that were created reflect lab capacity, efficiency, effectiveness, and capability—the four metrics that matter to program leaders and make it possible for them to manage labs more effectively.

Assessing the performance of the system integration labs not with ad hoc metrics valued only by technicians and engineers but with a standard set of metrics that matter to decision makers needing a holistic business perspective can lead to valuable manufacturing-environment benefits, such as the following:

- **Transparency**. With a clear, communicable set of metrics, leaders can quickly and accurately assess performance and capacity. In addition, fact-based, apples-to-apples comparisons will enable them to contrast the performance of one lab to that of others.
- **Cost savings**. Historically, cost advantages between labs have been hidden behind immaterial metrics. Now equal, meaningful metrics highlight current cost-saving opportunities.

Figure 4
Lab comparison across common metrics



Source: A.T. Kearney analysis

Figure 5
Example analytic framework for workload shift across integration labs

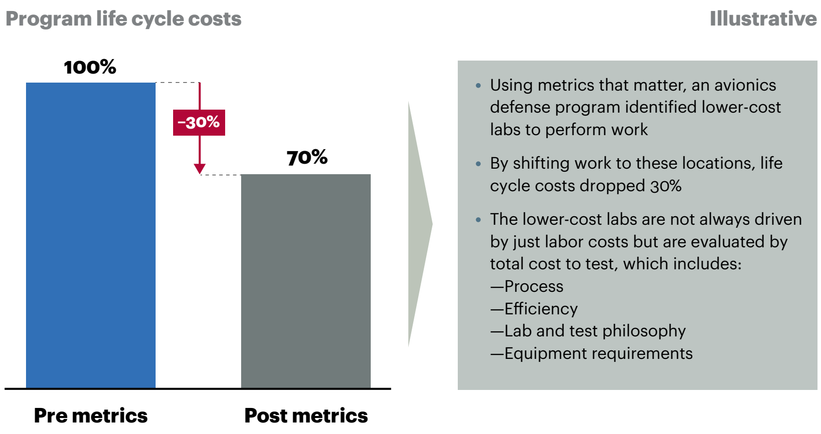
Demand calculation **Illustrative**

TP = test points
aTP = adjusted test points

				Lab Transitioning	
		Baseline	Period 1	Period 2	Period 3
Nominal demand	Test hours	200	200	200	200
Adjusted capacity	Raw capacity (test points/HR)	× 2 TP/hr	× 2 TP/hr	× 1.8 TP/hr	× 1.6 TP/hr
	First time right	× 90%	× 90%	× 90%	× 90%
Adjusted demand	Adjusted test points executed	= 360 TP	= 360 TP	= 324 TP	= 288 TP
Demand adjustment	Performance variance (test points)	N/A	0 TPs	36 TPs	72 TPs
	Hours surge required	0	0	23 hrs at 1.62 aTPs/hr[1]	50 hrs at 1.44 aTPs/hr[1]
	Revised test hours	200	200	223	250

[1] aTPs/hr: Adjusted test points per hour. Adjusted test points = raw test points / hour * first time right percent.
Source: A.T. Kearney analysis

Figure 6
Focusing on metrics that matter can reduce life cycle costs



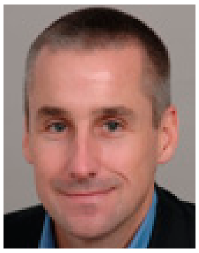
- Using metrics that matter, an avionics defense program identified lower-cost labs to perform work
- By shifting work to these locations, life cycle costs dropped 30%
- The lower-cost labs are not always driven by just labor costs but are evaluated by total cost to test, which includes:
—Process
—Efficiency
—Lab and test philosophy
—Equipment requirements

Source: A.T. Kearney analysis

- **Risk mitigation**. The metrics will take into account current and future lab capacity, allowing for more accurate estimates of cost and potential schedule delays.
- **Negotiations support**. The metrics will provide the facts on which the best negotiations are based and enable DoD leadership to accurately size and negotiate requirements for contracting labs.

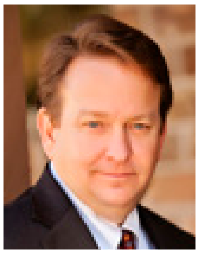
Moreover, with these metrics that matter, program leaders will have the solid measures they need to develop a full understanding of the labs' current level of efficiency, a starting point on which to base both minor and command-level decisions for the future and for determining the impact of those decisions—whether they are about adding capacity, reducing costs, hiring employees, improving throughput and quality, or similar issues. And as they make these decisions that will drive the effectiveness and savings of labs across the footprint, they will further strengthen labs' role in delivering the most advanced systems to U.S. weapon programs.

ABOUT THE AUTHORS



Christian Hagen is a partner in A.T. Kearney's Strategic Information Technology Practice and is based in Chicago. He advises many of the world's largest organizations across multiple industries, including government and defense contractors. He specializes in helping clients leverage software and information technology to increase efficiencies and gain competitive advantage. Christian has led several global studies for A.T. Kearney and authored over 60 published papers on low-cost competition, software engineering, e-commerce, technology innovation, and strategy.

E-mail: christian.hagen@atkearney.com



Steven Hurt is a partner in A.T. Kearney's Public Sector and Defense Services. Steve has worked with several of the USAF's highest-visibility programs to drive affordability in both software and hardware sustainment. Specifically, Steve has focused on should-cost analyses, business case analyses, contract negotiations, and developing business intelligence aimed at reducing cost.

E-mail: steven.hurt@atkearney.com

Andrew Williams is a principal in the Strategic Information Technology Practice and in Public Sector and Defense Services at A.T. Kearney. Andrew works with both commercial and government clients on some of the most challenging IT issues, including cost optimization, should-cost evaluations, infrastructure services, and large mega-vendor contract negotiations.

E-mail: andrew.williams@atkearney.com

REFERENCES

1. Ferguson, R. (2012). "A Project Dashboard." Unpublished Manuscript, Software Engineering Institute at Carnegie Mellon University.
2. Stark, G. (2011). "A Comparison of Parametric Software Estimation Models Using Real Project Data." CrossTalk – The Journal of Defense Software Engineering, January 2011.
3. Boehm, B. (1981). Software Engineering Economics. Englewood Cliffs, N.J., Prentice Hall.
4. Boehm, B. (2006). "Minimizing Future Guesswork in Estimating," IBM Conference on Estimation, Atlanta, GA, February 2006.
5. Jones, C. (2007). "Software Estimating Rules-of-Thumb." <http://www.compaid.com/caiinternet/ezine/capers-rules.pdf.>, March 2007.
6. Jones, C. (1997). Applied Software Measurement, 2nd Ed. McGraw-Hill, NY.
7. Rone, K. et al. (1994). "The Matrix Method of Software Project Estimation." Proceedings of the Dual-Use Space Technology Conference, NASA Johnson Space Center, Houston, TX, February.
8. J.W. Bailey and V.R. Basili. "A Meta-Model for Software Development and Resource Expenditures." Proceedings of the 5th International Conference on Software Engineering. New York: Institute of Electrical and Electronic Engineers, 1983.
9. ISBSG, International Software Benchmarking Standards Group. <http://www.compaid.com/caiInternet/ezine/ISBSGestimation.pdf>.
10. ISBSG, International Software Benchmarking Standards Group. <http://www.isbsg.org/isbsg.nsf/weben/Project%20Duration>.
11. McConnell, S. (2006). Software Estimation: Demystifying the Black Art, Redmond, WA, Microsoft Press.
12. International Function Point User's Group (IFPUG), Function Point Counting Manual, Release 3.1, 1990.
13. Jones, C. "Achieving Excellence in Software Engineering." Presentation to IBM Software Engineering Group, March 2006.
14. P. Oman. "Automated Software Quality Models in Industry." Proceedings of the Eighth Annual Oregon Workshop on Software Metrics (May 11-13, Coeur d'Alene, ID), 1997.
15. G. Atkinson, J. Hagemeister, P. Oman & A. Baburaj. "Directing Software Development Projects with Product Measures." Proceedings of the Fifth International Software Metrics Symposium (November 20-21, Bethesda, MD), IEEE CS Press, Los Alamitos, CA, 1998, pp. 193-204.
16. T. Pearse, T. Freeman, & P. Oman. "Using Metrics to Manage the End-Game of a Software Project." Proceedings of the Sixth International Software Metrics Symposium (Nov. 4-6, Boca Raton, FL), IEEE CS Press, Los Alamitos, CA, 1999, pp. 207-215.
17. Kemerer, C. F. (1987). "An empirical validation of software cost estimation models." Communications of the ACM, Vol. 30, No. 5, pp. 416-429.
18. Jorgensen, M., and Sheppard, M. (2007). "A Systematic Review of Software Development Cost Estimation Studies." IEEE Transactions on Software Engineering, Vol. 33, No. 1, January, pp 33-53.
19. Fenton N. E., and Pfleeger, S. L. (1997). Software Metrics: A Rigorous & Practical Approach, 2nd Ed., London, PWS Publishing.
20. Jorgensen, M. (2004). "A Review of Studies on Expert Estimation of Software Development Effort." Journal of Systems & Software, Vol. 70, No. 1-2, pp. 37-60.
21. Hagen, C., Hurt, S., Sorenson, J. "Effective Approaches for Delivering Affordable Military Software." CrossTalk – The Journal of Defense Software Engineering, Vol. 26 No. 6 (November-December 2013).
22. OEE Primer, <http://www.oee.com/calculating-oee.html>.
23. C.J. Bamber et al. (2003). "Cross-Functional Team Working for Overall Equipment Effectiveness (OEE)." Journal of Quality in Maintenance Engineering, Vol. 9 Issue 3, pp. 223-238.
24. J. Cohen, R. Ferguson & W. Hayes (2013). "White Paper: A Defect Prioritization Method Based on the Risk Priority Number." Internal White Paper, Software Engineering Institute at Carnegie Mellon University.