MEASURES FOR EXCELLENCE

Software

Quality Assurance

(SQA)

Of

Management Processes

Using

The SEI Core Measures

QUANTITATIVE SOFTWARE MANAGEMENT LTD

7 Rue Fenoux		41A Aynhoe Road
75015 Paris	Internet: qsm.europe@pobox.com	London W14 0QA
Tel: 33-140-431210	CompuServe: 100113,3364	Tel: 44-207-603-9009
Fax: 33-148-286249	www.qsm.com	Fax : 44-207-602-6008

Software Quality Assurance of Management Processes Using Core Measures

Introduction

Software Quality Assurance (SQA) without measures is like a jet with no fuel. Everyone is ready to go but not much happens. This paper deals with the SQA activities to ensure the right fuel is available to reach the destination, namely high quality. There are three processes, common to both development and purchasing, that enable organizations to be high flyers and reach quality in the stratosphere.

The role of SQA, as described here, is to confirm that core measures are used effectively by management processes involving technical and purchasing directors, project and purchasing managers and process improvement groups. They cover:

1. Benchmarking and Process Improvement
2. Estimating and Risk Assessment
3. Progress Control and Reporting

Process improvement enables the same amount of software to be built in less time with less effort and fewer defects. Informed estimating uses process productivity benchmarks to evaluate constraints, assess risks and to arrive at viable estimates. Estimates of the defects at delivery use the history from benchmarked projects and allow alternative staffing strategies to be evaluated. Throwing people in to meet tight time to market schedules has a disastrous impact on quality. Progress control tracks defects found during development in order to avoid premature delivery and to ensure the reliability goals are achieved.

Each process contributes separately to improving the quality of the final software product. We describe how the core measures are used in each process to fuel improved quality. Dramatic quality improvements are achieved by dealing with all three. (Ensuring the fuel is high octane).

SQA is defined as " a group of related activities employed throughout the software lifecycle to positively influence and quantify the quality of the delivered software." (Ref 1.) Much of the SQA literature relates to product assurance. This article focuses on process assurance and the core measurement data that supports all management levels.

The basic fuel elements are the Carnegie Mellon Software Engineering Institute (SEI) recommendations on core software measures, namely software size, time, effort and defects. (Ref. 2) An extra benefit is that the majority of the SEI-Capability Maturity Model (CMMI) Key Process Areas (KPA's) are met by assuring the processes use these measures. Criticisms leveled at large purchasing organization, typified by the U.S. Federal Aviation Administration (FAA -Ref 2), are answered by assuring the data is actively used in the three processes.

The first area, benchmarking and process improvement is a self-evident need for every software development organization concerned to demonstrate value for money and to reduce the software lifecycle time and effort as well as the number of defects. Benchmark results enable estimates to be made for new developments based on known process productivity and also provide the means to check these estimates against the benchmark history. Purchasing organization need to benchmark their software suppliers (including outsourcing suppliers) to establish contract baselines and build solid evidence of supplier on-going process improvement.

Software estimating and risk assessment is fundamental given the well-documented evidence of continuing software disasters characterized by cancelled projects as well as horrific cost and schedule overruns coupled with poor delivered quality (Ref. 10). Equally purchasing must evaluate proposal estimates quantitatively, compare supplier bids, establish a contractual baseline and ensure value for money as well as manage risk.

In-progress control and reporting of development progress gives continuous visibility to both developers and purchasers. Time to market pressure in many high technology domains' means that developers and purchasers alike require at all times to be confident the software delivery date, the expected cost and reliability will be achieved as planned.

Figure 1 illustrates these concepts. Benchmarking and process improvement quantification confirms that real commercial benefits are being achieved through improved development productivity. Project managers in development and purchasing ensure realistic estimates are made consistent with known constraints and the benchmarked process productivity. Equally important is to quantify the estimate uncertainty and risk.

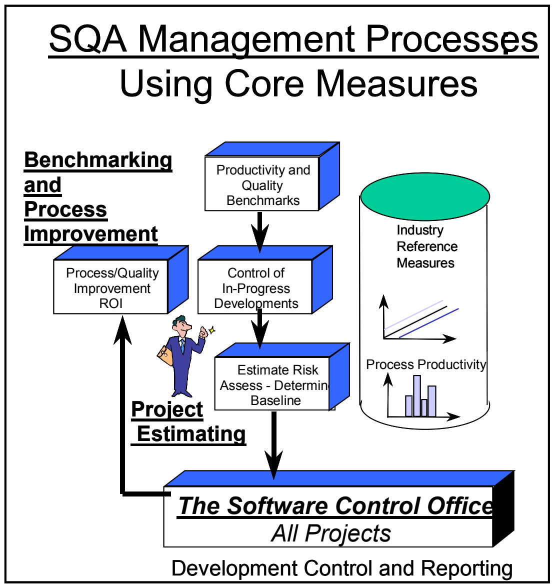

SQA Management Processes
Using Core Measures
Benchmarking and Process Improvement
Productivity and Quality Benchmarks
Industry Reference Measures
Process/Quality Improvement ROI
Control of In-Progress Developments
Process Productivity
Estimate Risk Assess - Determin Baseline
Project Estimating
The Software Control Office
All Projects
Development Control and Reporting

Figure 1: Software Management Processes

The objective is to agree a quantified development estimate baseline that is then used to control development. The Software Control Office (SCO) function provides high-level management control and reporting across all development projects. Each month this function collects basic progress data and independently assesses each project to highlight and report potential risks, delays, cost overruns as well as quality concerns.

The Benchmarking Process

Benchmarking is practical with minimal data, namely the time, total effort and software size for main development phase. Defects are an optional extra. This data is collected for recently completed projects. Experience shows this core data is assembled in about half a day for completed developments (Ref. 9). Purchasing requests the same data to build benchmark measures of supplier performance and uses the results to negotiate current and future developments. SQA verify this data is being collected and is used to build the benchmark database.

As developments complete a review examines the detailed monthly progress data collected by the SCO (see below). SQA participates in the review to verify the SCO has assembled the complete history for each development. The final data is added to the benchmark database of projects. Over time this growing database provides concrete evidence of development process productivity and quality improvement.

A regular procedure quantifies improvement benefits. At intervals, say every 6 months, a report is made showing benefits from recent projects through initiatives such as CMMI. This includes calculating the Return on Investment (ROI) based on productivity improvement and investments made to improve. A case study showing the result of managing a productivity program including the ROI is set out in Ref 6 Chapter 12. Purchasing look for similar evidence that suppliers are improving their process productivity at least in line with industry reference measures. This enables informed negotiation of new contracts.

SQA: Benchmarking and Process Improvement

- Is the minimum SEI Core Measurement data kept for all developments?
- Are reviews carried out for all completing developments?
- Is benchmarking being carried out every 6 months against:
  * Industry Reference Measures? * Internal Reference Measures?
- Are realistic process improvement objectives set with the forecast commercial benefits and Return On Investment (ROI)?
- Is the process improvement program planned, funded and actively supported by management?

The figures 2, 3,4 and 5 below illustrate how the core measurement data is used to benchmark. Details of these techniques and measures, including their engineering basis, are described in Reference 6.

Benchmark: Development Time (Months)
Versus Industry Trend Lines

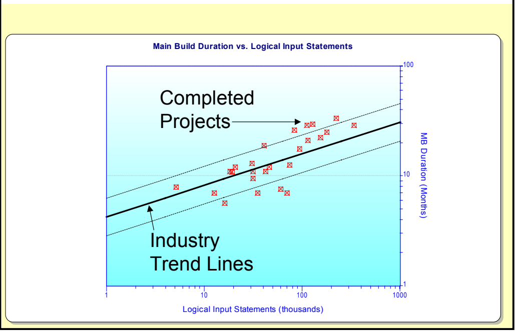


Figure 2: Development Time versus Size

Benchmark: Development Effort (Person Months)
Versus Industry Trend Lines

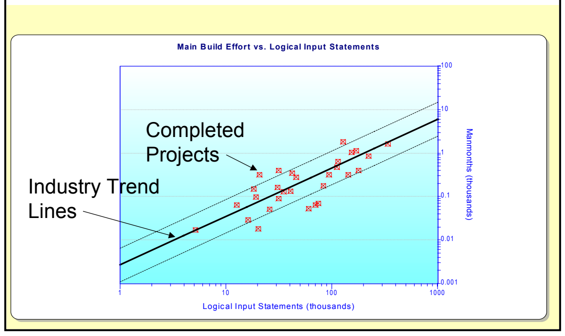


Figure 3: Development Effort versus Size

Benchmark: Number of Projects Versus Process
Productivity Index (PI) & Industry Average

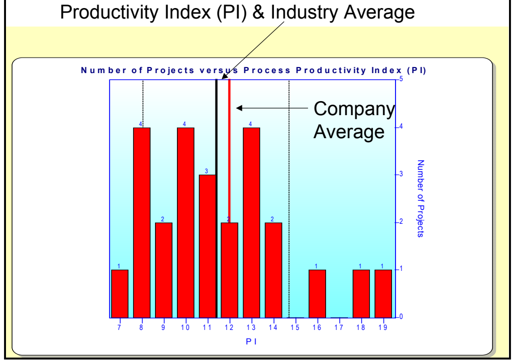


Figure 4: Process Productivity Distribution

Benchmark: Number of Projects Versus Time Pressure
(Manpower Buildup Index = MBI) & Industry Average

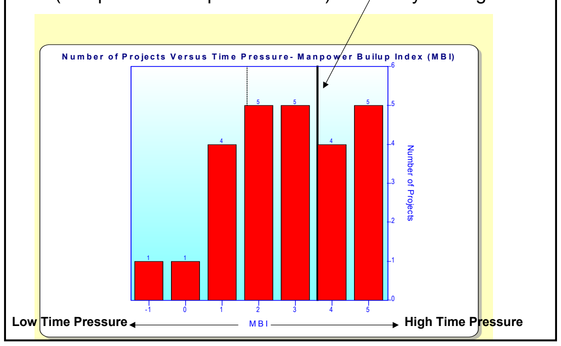


Figure 5: Time Pressure (MBI) Distribution

The Estimating and Risk Assessment Process

In a development group SQA is performed on the documented estimating procedure to check input data is used that quantifies:

- The software product size and uncertainty
- The development process productivity
- The development constraints: time, effort, staff, and reliability
- The risk levels for each constraint

Software acquisition requires the supplier to provide this data using a formal questionnaire. Software size is quantified using the estimated size range. The range reflects specification uncertainty that reduces as progress is made through the feasibility and specification phases. Greater detail is practical as feature specifications are refined. Each software module is estimated in terms of the smallest, most likely and largest size. This size range uses the most practical sizing units such as logical input statements, function points or objects (Ref. 5).

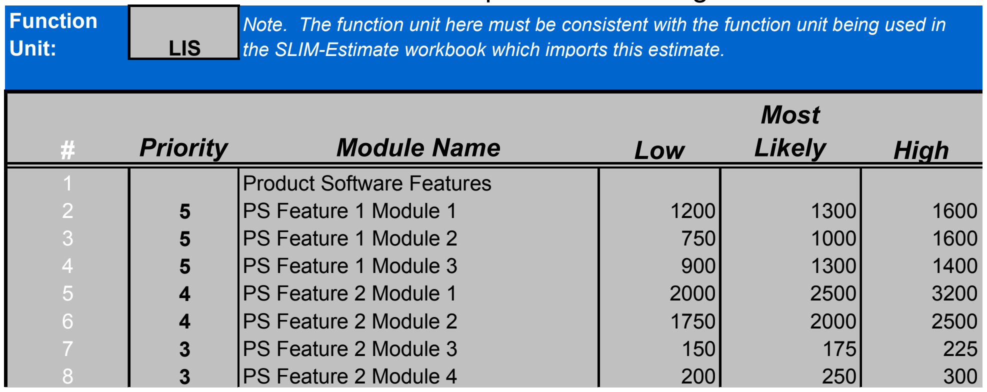
SQA: Development Estimate and Risk Assessment

- Are the Features identified, prioritized and mapped to software modules?
- Is each module estimated at less than 3000 statements with the <u>size range</u> to quantify uncertainty?
- Is the assumed process productivity consistent with completed projects?
- Are the development constraints clearly stated and prioritized with agreed risk levels?
- Are alternative estimates logged and documented?
- Is a baseline estimate agreed consistent with size, process productivity, constraints and completed projects?

Size estimating guidelines require all modules be identified that are to be modified or are new. The guidelines specify the largest "most likely" size required- for instance 3,000 Logical Input Statements (LIS). This ensures sizing is made at a detailed level so that complete identification is made of changed and new modules. In practice this detailed module breakdown is required to allocate work to individual programmers.

The size and uncertainty of the full development is calculated using statistical techniques to give the mean size and standard deviation. Features and their corresponding modules are prioritized. This allows the rapid evaluation of alternative estimates depending on the features included.

An example of module sizing with feature priorities is shown in Figure 6. Once development begins this agreed baseline is used to evaluate the impact of proposed requirements changes.



Function Unit: LIS

Note. The function unit here must be consistent with the function unit being used in the SLIM-Estimate workbook which imports this estimate.

#	Priority	Module Name	Low	Most Likely	High
1		Product Software Features			
2	5	PS Feature 1 Module 1	1200	1300	1600
3	5	PS Feature 1 Module 2	750	1000	1600
4	5	PS Feature 1 Module 3	900	1300	1400
5	4	PS Feature 2 Module 1	2000	2500	3200
6	4	PS Feature 2 Module 2	1750	2000	2500
7	3	PS Feature 2 Module 3	150	175	225
8	3	PS Feature 2 Module 4	200	250	300

Figure 6: Feature Priorities, Modules and Size Range

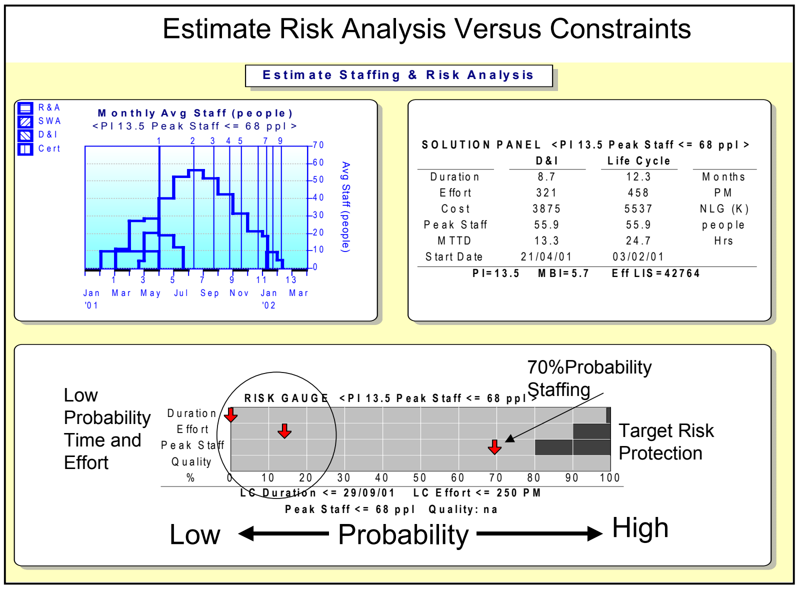
The process productivity for the estimate uses the benchmark results from completed projects wherever possible. If no benchmark measures exist then industry measures are used, keeping in mind that more uncertainty and risk will result.

The time, effort, resources, costs and reliability constraints for the development are risk assessed taking in to account the quantified uncertainties such as the size range. Each constraint is associated with a risk level and estimates are evaluated against these risk levels. An example is shown in Figure 7.

Frequently it is found that specific constraints cannot be met since the risk is too high. The estimating procedure evaluates alternatives, each of which is logged and documented. This may mean allowing additional time, adding staff and/or reducing features (size). The alternative "What If" estimates document how the final baseline plan is determined, risk assessed and agreed.

Purchasing requests the estimate data using a formal software tender questionnaire. This allows purchasing managers to evaluate the supplier development proposal and negotiate and risk assess the final contractual baseline. Figure 8 shows an example of alternative estimates. In a purchasing organization this can be an evaluation of separate competitive proposals.

SQA in both development and purchasing confirm the process above takes place for each estimate and that documented outputs support the baseline plan. This plan then becomes the contractual basis for the development. The plan also includes major milestones within the development covering such activities as major software design reviews, increment deliveries, all code complete, start and completion of integration, test case plans and all key activity dates unique to the development.



SOLUTION PANEL <PI 13.5 Peak Staff <= 68 ppl >

	D&I	Life Cycle	
Duration	8.7	12.3	Months
Effort	321	458	PM
Cost	3875	5537	NLG (K)
Peak Staff	55.9	55.9	people
MTTD	13.3	24.7	Hrs
Start Date	21/04/01	03/02/01	

PI=13.5 MBI=5.7 Eff LIS=42764

Figure 7: Estimate Risk Assessment versus Constraints

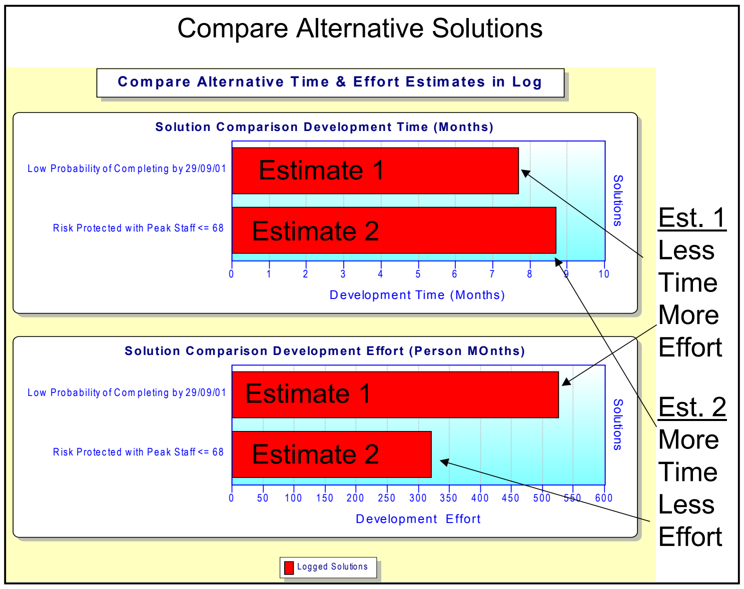


Figure 8: Comparison of Alternative Logged Estimates

Software Control Office (SCO) Process

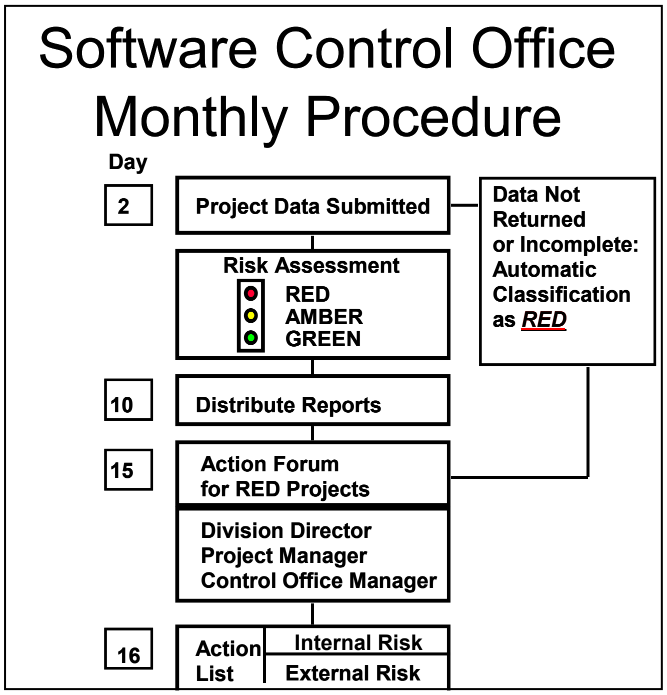


Figure 9: The Software Control Office Process

A pragmatic procedure is followed each month to ensure continuous visibility in each development by assessing and reporting progress and risk. The management process, illustrated in Figure 9, is common to both development and purchasing organizations. The procedure is operated by a separate function, the Software Control Office (SCO). SQA is performed on the SCO control procedure and the inputs and outputs.

SQA verifies the monthly progress data is returned for each software project. The accompanying box lists this data. Note the data is basic to managing each development. If the data cannot be provided the development is out of control. The data is used independently by the SCO to detect variance against the baseline plan and to report to all management levels. Actions are taken and followed up whenever risks are detected. A forecast is made of the outstanding development work if slippage is detected. The forecast becomes the revised development plan if agreed by the senior manager. SQA includes checking that the dates for the procedure sequence are observed and that key managers actively participate.

SQA: Progress Data and Control

- Monthly Progress Data
  - Staffing- people allocated to the project
  - Key milestones passed
  - Each module status- is it in design, code, unit test, integration or validation?
  - Program module size when code is complete
  - Total code currently under configuration control
  - Software defects broken down in to critical, major, moderate and cosmetic
  - The number of completed integration and validation tests.
- Check variance analysis performed
- Ensure completion forecasts made
- Confirm audit trail kept of plans and forecasts
- Verify change request impact documented
- Verify the correct management levels participate
- Confirm control cycle operates as specified
- Check actions are followed up and recorded

Every month a management summary is produced for all current and future projects summarizing the risks in each development. This summary is illustrated below, Figure 10, showing the report for each project produced in a purchasing organization (Ref. 3). Traffic lights are set at Red, Orange or Green to summarize the current risk status. Red projects get close management attention and follow up action through the operation of the SCO.

Change requests during development are sized and assessed in terms of their impact on the agreed baseline. Acceptance of the change requests leads to a revised plan for the development.

Each plan change and forecast is logged to provide a complete audit trail for the development.

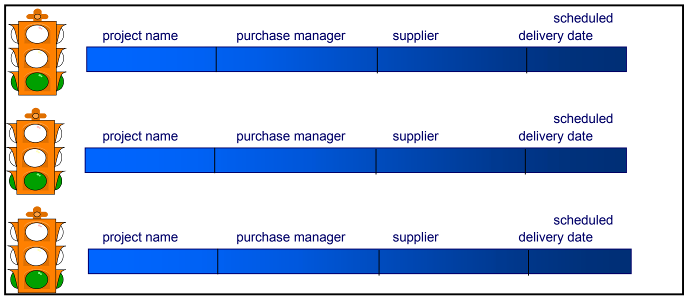


Figure 10: Monthly Summary Risk Report

Software Quality Assurance: Product Assurance: Post Implementation Review

SQA: Defects and Completion Measures

- Confirm monthly history is complete
- Review defect data and defect tuning: Critical, Major, Minor Defects
- Review forecast of remaining defects
- Assess Mean Time To Defect (MTTD) at delivery
- Check reliability criteria met for acceptance- avoid premature delivery

SQA frequently ensures product delivery is made with few software defects remaining. The final project review evaluates the defects to date and confirms high reliability in the software product for delivery. Defect data collected each month is analyzed and used to forecast the number of critical and major defects remaining. This avoids the premature delivery of the software and all its attendant problems (Ref. 7).

Using the core measurement data provides visibility and control each month throughout development. The data provides a full history when the project completes. This history is invaluable to understand how the project performed and to add to a growing database of development performance.

Notes are also kept throughout development by the SCO. These are used to investigate the history in the development and to learn lessons for future projects*.*

SQA Core Measures Summary

The SQA function validates the major management areas by auditing the core measures, their use by the management processes and making sure the processes are followed. Below is a summary of the management areas where SQA is performed and the likely owner of the specific process.

SQA Performed On	**Owner**
Benchmarking and Process Improvement	
Metrics Repository	Software Engineering Process
Completion Review	Group (SEPG) or Software
6 monthly Industry Benchmark Comparison	Process Improvement (SPI)
Process Improvement Benefit	/Purchase Manager
Development Estimate/Risk Assessment	
Feature Sizing and Priorities	Software Project Manager
Development Constraints	/Purchase Manager
Risk Assessment of Alternatives	
Development Estimate Baseline	
Development Control	
Monthly Progress Data	Software Control Office:
Variance Analysis versus Baseline	Development/Purchasing
Progress Risk Evaluation and Reporting	
Change Assessment and Re-planning	
Audit Trail: Plans and Forecasts	
Management Involvement and Actions	

Conclusions: SQA of Management Processes : Core Measures

SQA is only practical in the three key management processes described here by using measures. Each process directly impacts the final software product quality. For instance the management decision with respect to development time significantly influences the number of defects (Ref. 6). Research shows that a short development time means a large increase in staff, effort and costs resulting in many more defects. Conversely allowing development to take just one or two months longer substantially reduces all these values by as much as 50% (Ref.6).

Hence it is vital these management processes operate using quantified data. The core measurement data enables all the SQA objectives to be met in these processes (Ref. 8). We find the major challenge is to set up, operate and then SQA the Software Control Office. Here the key to success is to show the measurement data provide all managers with firm evidence of potential risks, first related to the development baseline estimate and then each month as the development progresses.

By introducing the SCO function the project or software acquisition managers ensure the measures are actively used to reduce risks and get top management attention and action. These managers are busy people. Only by continuously showing the data is of practical benefit to the organization is it possible to motivate them. SQA plays a vital role to ensure these management processes operate and actively use the core measurement data. The quality plan in every development group should include these topics.

On a personal note the author first introduced these techniques as part of SQA some 20 years ago on first contacting Lawrence Putnam to understand his engineering analysis of software development. At that time the main technique developed by Putnam, derived from empirical data from software projects (Ref 6), related to development estimating and risk assessment. Putnam's quantified techniques now extend to benchmarking and software development control. The scope for SQA to deal with benchmarking, quantifying process improvement and development control is likewise enhanced.

Jim Greene is Managing Director of Quantitative Software Management Europe in Paris, France: telephone 33-140431210; fax 33-148286249. He has over 35 years experience in software engineering, with a particular interest in management methods used by development and purchasing organisations based on the quantification of software development.

Ref 1: U.S. Air Force Software Technical Support Centre http://stsc.hill.af.mil: Software Quality Assurance -A technical Report Outlining Representative Publications April 2000 Revision 6

Ref.2: The SEI Core Measures Anita D. Carleton, Robert E. Park and Wolfart B. Goethert The Journal of the Quality Assurance Institute July 1994

Ref. 3: Geerhard W. Kempff "Managing Software Acquisition" *Managing System Development* July 1998 Applied Computer Research Inc. P.O. Box 82266, Phoenix, AZ, USA.

Ref. 4:GAO Report to the Secretary of Transportation: Air Traffic Control GAO/AIMD-97-20

Ref. 5: J. W. E. Greene "Sizing and Controlling Incremental Development" *Managing System Development* November 1996 Applied Computer Research Inc. P.O. Box 82266, Phoenix, AZ, USA.

Ref. 6: L.H. Putnam "Measures For Excellence: Reliable Software, On Time, Within Budget:" Prentice Hall New York 1992

Ref. 7: J.W.E. Greene "Avoiding the Premature Delivery of Software" QSM paper see www.qsm.com 1996

Ref: 8 J.W.E. Greene "Software Acquisition Using the SEI Core Measures" European Software Engineering Process Group Conference Amsterdam SEPG 2000

Ref. 9 John Tittle Computer Sciences Corporation "Software Measurement and Outsourcing" Presentation at the QSM User Conference October 2000 see www.qsm.com

Ref.10 The Standish Group Chaos Report 1995

For further information on the practices described here, please refer to Lawrence H. Putnam and Ware Myers, "Industrial Strength Software: Effective Management Using Measurement", IEEE Computer Society Press, Los Alamitos, CA, 1997, 309 pp.