The Association of System Performance Professionals

The **Computer Measurement Group**, commonly called **CMG**, is a not for profit, worldwide organization of data processing professionals committed to the measurement and management of computer systems. CMG members are primarily concerned with performance evaluation of existing systems to maximize performance (eg. response time, throughput, etc.) and with capacity management where planned enhancements to existing systems or the design of new systems are evaluated to find the necessary resources required to provide adequate performance at a reasonable cost.

This paper was originally published in the Proceedings of the Computer Measurement Group's 2004 International Conference.

For more information on CMG please visit www.cmg.org

A PERFORMANCE PROCESS MATURITY MODEL

Michael Maddox
MCI
Colorado Springs, Colorado

Computer performance science is very well developed, yet the state of the practice somewhat lags the state of the art. This paper presents a model of computer performance process maturity including signposts of each maturity level, suggests types of benefits accruing from advancing in maturity, and offers basic guidelines for advancing from level to level.

1 Scope and Purpose

This paper attempts to connect two quantitative disciplines: computer performance discipline and performance of the enterprise. It assumes that the two disciplines are working in the same direction. Yet while managers and performance workers will agree that better computer performance generally supports enterprise goals, few disciplines exist to align the two.

This paper enumerates performance techniques, but does not dwell on how they are done or present new ones. Rather, it describes how they fit into an overall enterprise strategy with increasing effectiveness. This paper describes a logical progression from chaos, uncertainty, and risk, to harmony and efficiency. For each level of maturity, it presents the characteristic behaviors of that level, the improved orderliness and efficiency of that level over preceding levels, and a few suggestions on how to advance to the next level.

Finally, this paper presents ideas for taking the performance process maturity model itself to the next stage of evolution.

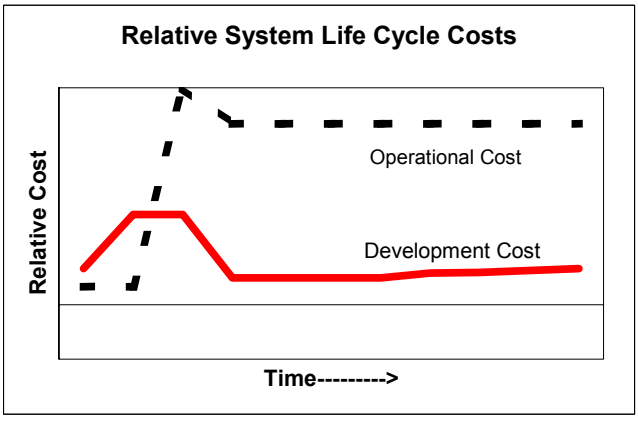
2 Background: How Does Performance Fit Into Larger System Contexts?

This paper assumes there are three key functions in the system life cycle; they may differ among enterprises:

- The **business management** function identifies the need for a new computer-based system, justifies it, provides requirements, and arranges funding.
- The **development** function develops the system to the requirements supplied and maintains it once deployed.
- The **operations** function deploys the system and uses it for business benefit.

Let's put system life cycle costs and time into perspective for just a moment. Figure 1 (below) admittedly reflects projects and systems in the author's experience, not a rigorous literature search. (Specifically, the terms Operational Cost and Development Cost in this chart denote the intuitive ideas of the terms, not to elements of formal, rigorously defined cost models.)

Figure 1 System Life Cycle Costs



The idea here is simply that a computer system is built (costing money to develop), then it is used to generate business benefit (costing more money to deploy and use). The amount spent to use a computer system and get a business benefit from it often far exceeds the amount spent to develop it.

Considerable thought and study have been devoted to the first phase of the life cycle, development (including requirements definition, analysis, and so on). For example, the Software Engineering Institute's (SEI) Capability Maturity Model (CMM) comprises a key Department of Defense yardstick for measuring process optimization. Yet, CMM and other software process improvement (maturity) models concern themselves with only a fraction of the total amount spent to develop, deploy, and use a computer system.

Common sense suggests that we look at operational processes and costs—including performance-related activities—every bit as carefully as we do development processes. This paper assumes a holistic view of the application environment—spanning requirements definition, development, and operations—and presents a means for measuring performance process maturity through the life cycle.

3 What Are the Levels of Performance Process Maturity?

The author proposes five levels of performance maturity, numbered from 1 to 5. All levels have these common characteristics:

- The levels are cumulative. Performance activities and processes practiced at level 2 are retained and enhanced at level 3, and so on through higher levels.
- Different applications may exhibit different maturity levels—although the single corporate or division culture that gives rise to them may result in similar maturity levels for most systems.
- Some level of learning and feedback is applied as work progresses. Organizations at higher levels of maturity apply more effective, more strategic feedback.

3.1 Maturity Level 1: Fire Fighting

At Level 1, developers create systems with little awareness of operational considerations, including performance. The requirements they are given specify only the most basic performance needs, if any at all. If performance issues are exposed in pilot or early deployment, they are addressed by "tuning"—minor adjustments to program logic yielding only incremental improvements. Systems delivered for production are effectively "thrown over the wall."

Servers may be specified, purchased, and installed with little understanding or quantitative science applied to their sizing, leading to unexpected outages followed by emergency (expensive) upgrades or replacements or even application rework. Rework due to an application which is too slow to use can be expensive (involving cost overruns) and entail significant delays.

Changes in user population or user application sets come as surprises to operations staff, with downtime or dramatically poor performance as a result.

There is a strong emphasis on troubleshooting and a reactive mode of behavior, both primarily in the operations domain. A support team member with little performance expertise may be assigned to "run a quick PerfMon" on a Microsoft Windows server or run other spot checks with ad hoc-type tools. Each new crisis generates another special study, and some guidance may be derived for preventive actions, but there is little attempt to generate strategic value or to plan for long-term, stable processes.

At the management level, there is little understanding of how system performance contributes to the success of the enterprise—only the awareness that downtime (an extreme case of poor performance) costs money. There is little performance leadership. Performance is a dark art at this level of maturity.

3.2 Maturity Level 2: Monitoring

At level 2, a performance worker sets up some level of automation to collect performance data from systems in production, ideally 24x7. There is some effort to deal systematically with resource measurements such as CPU utilization, I/O bandwidth, memory availability, or disk space that exceed established thresholds. The performance worker may publish reports regularly, but the management audience may still consider the data and what it means to be beyond their interest or expertise.

The systems which receive monitoring attention have been rationalized to at least some extent, bearing in mind the number of users and the resulting cost of downtime and poor responsiveness.

Application systems are still subjected to little performance scrutiny prior to deployment. As a result, unexpected levels of user populations or system loads can still disrupt response times and stability. Efforts to fix or prevent performance defects may be limited to operating system (OS) or hardware configuration adjustments, system software updates, and the like.

The performance worker may be able to use a packaged monitoring system, such as BMC Patrol Perform, Heroix eQ Management Suite, or others in that vein. Or, alternatively, the performance worker

may assemble a system from free components, linked by scripts and OS-level task scheduling. The monitoring system might catch key out-of-threshold measurements. There are two possible levels of alerting:

- Near-real time, providing operational support teams with information needed to tackle anomalies within 15 minutes, for example.
- Batch, providing more tactical or somewhat strategic support (with respect to operations only). In the general case, it tells team members that something happened yesterday or last week that bears closer inspection.

Despite what might be a very large investment in packaged software or a tailored development effort, though, performance process definitions that span organizational boundaries may be limited or nonexistent. (That is, "a tool is not a process.")

Capacity planning is limited to systems for which there exist performance data already collected, or can be collected in a special study. There may be little formal capacity planning expertise, modeling, or other rigorous performance methodology applied.

Response time data may be collected, giving rise to three sublevels of maturity we could term 2A, 2B, and 2C:

- Level 2A – No response time measurements available.
- Level 2B – Response time measurements are available from commercial, off-the-shelf (COTS) software and need only be enabled by operations and analyzed.
- Level 2C – Response time collection is coded into the application and development or operations team members analyze the resulting data. Application-generated response time measurements may supplement that available from COTS software.

In 2B and 2C, response time data are collected on a full-time basis, perhaps on a random sampling basis to keep data volumes manageable.

Response time measurement coded into an application represents a greater investment in the performance infrastructure and thus rates a higher maturity level. (Note that response time data can be used in two ways: first, as a direct measure of system responsiveness and second, analyzed for arrival rates, volumes, and trends over time.) The collection of application response times allows somewhat more detailed analyses, perhaps even detailed modeling of CPU and I/O resource utilizations, but the effort has more in common with an autopsy than with disease prevention.

Workload measurements such as transaction counts may be largely ignored, as the emphasis is more on resource measurements.

Detailed system configuration data, consisting of CPU counts and clock rates, memory complement, and logical and physical disk volume sizes and RAID configurations support performance analyses. (For Microsoft Windows/Intel-based systems, a systems management tool like Compaq Insight Manager makes this job practical for large server inventories.)

Performance leadership may come from a manager or a performance worker.

3.3 Maturity Level 3: Performance Optimizing

At level 3, performance evaluation and planning are baked into the development process. As Dr. Connie U. Smith [SMIT02] observes, developers "build performance into systems rather than try to add it later." Developers and performance workers tackle performance requirements with a full array of methods and tools. Here is a brief sampling:

- Ideally, a Software Performance Engineering (SPE) model predicts system responsiveness and resource contention from the moment even a "straw-man" design is advanced. The SPE model enables developers to predict the compliance of the system with performance requirements from inception to pilot and production. The SPE model is adjusted and refined as more details of the design and implementation emerge. Designers and developers receive and incorporate feedback from ongoing performance analyses. An SPE model incorporates both a software behavior model (resource times consumed, loops, etc.) and a system execution model (e.g., queuing network model). Dr. Smith offers a tool called SPE·ED meeting these specifications.
- A response time budget breaks down the timing of complex or multi-layered applications to establish internal processing time limits. The response time budget is established early in the project and refined as components are developed and their behavior becomes better defined.

- Integrated development environment (IDE)-type profilers evaluate execution paths and path lengths. Developers use similar features of computer aided software engineering (CASE) tools or Structured Query Language (SQL) servers which analyze path lengths, I/O counts, and other performance behaviors. Specialized mainframe-oriented tools such as MXG and MICS inspect application performance behavior. An I/O trace utility captures input/output patterns and timings and device utilizations. OS-specific tools capture a wide variety of performance measurements from test runs.

- The application is developed to use the Application Response Measurement (ARM) application programming interface (API) to capture response times in execution. A monitoring system such as HP's Response Time Workbench supplies an agent to support the industry standard ARM API, collecting and analyzing response times. (ARM itself is not specifically essential for response time analysis. Yet multi-layered system architectures—servers behind servers—present special challenges. ARM level 2 and later are uniquely designed to address those challenges.)

- With or without ARM, the identities of back-end transactions (those sent to a legacy mainframe system, for example) are well known and their response times are logged. (It is not enough to know that mainframe application XYZ is responding slowly; one must be able to supply XYZ's maintainers with the exact identities of errant transactions to support resolution efforts.)

- Volume tests, driven by a tool such as Mercury Interactive's LoadRunner or Microsoft's (free) Web Application Stress tool, evaluate the behavior of the application under higher volume conditions and confirm resource utilizations predicted by design-level models.

- A network "sniffer" confirms the application's network behavior, checking data volume, round trip count, and network-level response times.

Another key feature of level 3 is capacity planning. At level 3, capacity planning is formalized and uses a well-refined methodology, supported by a specialized capacity planning tool such as BMC Patrol Predict (formerly known as Best/1). The capacity planning tool is fully supported by data collected on defined workloads (a baseline). There is a well-established process for business measurements (expected changes in business volume, etc.) to feed the capacity planning activity.

A performance worker regularly publishes performance reports, using a high degree of automation. The reports use graphics, hyperlinks, and text formatting effectively to communicate with a wide audience. Performance measures are documented with clarity, affording both performance-knowledgeable and those with an interest in performance the ability to read and interpret the figures presented.

Level 3 performance workers encompass performance specialists, architects, and developers. Why? Performance must be ingrained in the development process and is the responsibility of all involved, not just the charter of a specialist.

Performance requirements given to developers may or may not comprehensively support business goals. Still, the performance requirements should include the following elements at a minimum:

- The workload, expressed as a number of users with an average think time, or a transaction arrival rate, such as "100 users and a 30-second think time."

- The workload profile or mix, expressed as percentages of specific work unit (transaction) types, such as "80% Query A, 10% Query B, and 10% others."

- Response time goal stated as a percentile, such as "95% of responses within 1.0 second." It is impossible to guarantee that fully 100% of transactions will be processed within a given span of time; stating a response time goal as a percentile fits better with the reality of performance analysis and system behavior.

- Anticipated changes in user counts or transaction loads over time, such as "25% growth is expected over the next three years."

- Expectations of how the application will be deployed, e.g., in a LAN (local access) or a WAN (geographically distributed) scenario. Addressing this as part of the requirements is critical because some applications are "chatty," making many server round trips per user interaction. Chatty applications may be unusably slow in a WAN scenario.

- Specific monitoring requirements which will be used to verify compliance with response time requirements, e.g., response time logging.

Each project's successful pilot and full deployment is followed by a post-deployment analysis (PDA) session. How closely does the system come to response time targets, resource requirements, increased user productivity, reduced training or support costs, and other goals? Each outcome stated in the business analysis case (justification for funding) needs to be evaluated against its realization. Lessons learned from the PDA are non-judgmentally applied as process refinements for future systems.

Finally, there exists awareness of and a process for dealing with business changes and how they impact system performance and resource requirements. That is, business management understands that a new (added) application cannot be simply dropped onto a server or desktop without some level of preparation. For its part, operations understands the need to deploy new functionality quickly and has an ongoing protocol with other functions to support business changes such as new applications appended to an existing suite.

Systems that are developed at maturity level 3 perform predictably within the performance limits specified, when used at the volume limits specified in the requirements.

Performance leadership and/or support has moved up to the director or vice presidential level.

3.4 Maturity Level 4: Business Optimizing

The primary improvement made in level 4 is that the contribution of the system to effectiveness of the business is understood much more fully, especially in (but not limited to) these areas:

- User productivity with the application (call duration, telemarketing sales per hour, and so forth) is well understood. Application behaviors and attributes (such as screen design) that impact user productivity are well understood and quantified.

- In general, the business value of the system—not just the benefit to user productivity—is well understood.

- Proposed changes to the system are evaluated thoroughly for their impact on user productivity and resource utilizations, building on a baseline of data collected in production (for example, screen flow modeling).

- Tradeoffs between system responsiveness, user productivity, hardware investment, and system lifetimes are well understood and rationalized.

- The complete costs of a system are measured and well documented from end to end, including training, user learning curves, help desk tickets, and so forth.

A wide range of coordinated skills (e.g., a multi-disciplinary team) is needed to reach this level of maturity, because it encompasses performance, human factors, management, and systems analysis domains, to name just a few.

It is difficult to provide exact formulas for this level of maturity, because it borders more on management science than computer science. Systems (at both the business and computer levels) that justify the detailed level of attention of level 4 will have both high scale (many users) and high concentration (frequency of use). For example, a call center application may represent both high scale and high concentration, where a travel expense reporting application may represent high scale, but low concentration.

At level 4, a process-oriented culture is emerging. Enterprise goals are visible to all and form a yardstick for measuring design and performance decisions. Likewise, an enterprise architecture is emerging. Optimization strategies apply now not just to individual application systems, but to groups of applications. Costs are being minimized across wide segments of the enterprise.

Points of interaction between business systems and computer systems make a fertile field for instrumentation and process optimization. Capitalizing on those points of interaction will require a high degree of cross-disciplinary cooperation and understanding (indeed, patience), but especially in high-scale/high-concentration situations, significant cost savings may await those who take on the challenge.

Performance leadership at levels 4 and 5 comes from the executive suite.

3.5 Maturity Level 5: Process Optimizing

At level 5, executive management understands the benefits of performance and process optimization fully. The focus in this level is to extend the benefits still further by:

- Closely examining costs versus profit possibilities where computer systems are involved.

- Rationalizing the benefits to be gained from each potential optimization against the costs of achieving that optimization, e.g., looking at return on investment (ROI).

This maturity level employs management science almost exclusively, yet the potential contribution of system performance is not ignored, where "systems" are examined thoroughly at all levels, from disk drives to balance sheet. The focus is on discovery and prevention of more and more subtle performance defects in systems of wider scope.

A process culture ensures visibility of enterprise goals to all. Everyone measures his or her efforts against process effectiveness, elevating process concerns to management when necessary. Process concerns raised in this way are evaluated at the appropriate level and the results communicated back to the originator.

At level 5, an enterprise architecture has been rationalized and performance of member applications working together has been optimized. In other words, performance optimization extends beyond individual application systems and the business systems built on them. Performance optimization is being achieved and improved across broad sectors of the enterprise.

4 What Other Progress Is Made Through Maturity Levels?

There is a wide variety of factors which show improvement as an enterprise moves to higher levels of maturity. This section surveys a few of them.

4.1 Risk Sources

At level 1, a system is at risk from ordinary use, since the system's performance behavior is not well understood. Surprises are a regular feature of working life at level 1.

At level 2, a body of understanding builds up over time, generating a slight degree of confidence. Yet since the performance characteristics of the system were not planned, they are not predictable. Unusual circumstances such as holiday transaction volumes can still disrupt performance expectations and produce instability and downtime.

At level 3, the performance characteristics of the system, as well as workload, are better understood. The main source of risk is workload outside the envelope specified in the requirements statement.

At level 4, the impact of the computer system on the larger, business-level system is much better understood. System behaviors affecting user productivity and other business measures—indeed those very measures themselves—are captured, reported, and analyzed for ways to produce still more improvement. At levels 4 and 5, there are very few sources of risk.

4.2 Excitement Level

At level 1, the system's performance behavior is uncertain, producing downtime, instability, and poor user productivity. Management and their teams' stress levels remain high, even between crises. One never knows which system will fail next, and in which location. Crises bring out heroism in individual workers searching for ways to stand out from the crowd. Workers are rewarded for outstanding efforts, yet there may be only a limited effort to prevent the circumstances that led to heroic efforts.

At levels 2 through 4, the system's performance behavior becomes increasingly predictable. Opportunities to respond to crises become fewer and farther between, simply because there are fewer crises.

At level 5, attention is on the performance of the larger system. Risks are lower and so is the stress level.

4.3 Costs, ROI, and ROI Time Frame

At level 1, every performance incident is a one-off; there is little value retained on a systematic basis. A team member finds a problem and fixes it. While there may be a policy that extends a found solution to other machines running the same application or OS or otherwise share common features, there is little organizational learning. Each problem seems very different and the underlying causes are not studied or understood well. The cost expended to resolve a problem is all but thrown away; it is difficult to leverage or reuse it for future benefit. The scope of a performance effort at level 1 is very narrow, limited to a single machine with a single problem—at least, that's the way it seems.

At level 2, production monitoring starts to reveal common issues. Problems exposed with one machine or OS or one type of application can be generalized somewhat. Learning from a problem recognized and resolved is applied (invested) for the longer term. That investment may take longer to produce benefits, but those benefits tend to last longer, too. Organizational learning has begun on a limited basis, but its scope may be largely limited to the operational sphere. A few lessons learned in operations may filter through to the development process, but only on an informal basis.

At level 3, business management and development management have started to invest in the longer-term future. Both have a better understanding of costs and how a small amount invested early in the development process benefits the whole enterprise for the life of an application. Lessons learned in the operational sphere are applied consistently to—and invested in—all phases of new-system development.

At level 4, enterprise management—those in the executive suite—not only have a clear vision of the power of performance management over the system life cycle, but they invest in the long-term future of a business-level system. Emphasis is on controlling costs in the long term, but each investment is also scrutinized for ROI. Costs of business benefits from process instrumentation (measurements of user productivity, for example) are compared to the benefit that will be derived, guiding decision making.

At level 5, the process of business optimization itself is monitored. The results of each process instrumentation decision are monitored for effectiveness and accuracy. Attention turns to perfection of the process and the longer-term future. Business optimization decisions are documented, monitored, and analyzed in hindsight. Processes and communication channels are adjusted as required to produce more effective processes. Process and policy changes are planned and their longer-term effects are understood well.

4.4 Alignment of Goals and Consistency of Measurements

Alignment of goals is about both cooperation and trust. All three improve with increasing maturity.

At level 1, each division which handles a portion of the system life cycle has its own agenda and its own goals. Traceability of each division's contribution to enterprise success—other than actual budget numbers—is limited. Because alignment of division goals is poor, divisions may even work at cross purposes; one division may succeed in advancing its agenda at the expense of another's. The enterprise as a whole loses, but in a way that is hidden and produces hidden costs. The measurements captured from various applications and systems have no pattern, no unifying intent or vision.

At level 2, a few participants try to seek some order in the madness. Monitoring produces numbers which can be studied and produce optimizations with limited effect. There is still little uniformity in the monitoring (capture of measurements) which is done, because the monitoring leverages only what is easily available; the choices of monitoring points to create were not decided with the full life cycle (much less enterprise goals) in view. In fact, some key applications may have almost no measurements captured at all. Overall, alignment of organizational agendas from a performance perspective (system level or business level) is still poor.

At level 3, business management, development, and operational teams are better at aligning their goals with the needs of the enterprise. Business management has a more complete understanding of factors at the system level that affect business-level performance, as well as a clearer picture of the opportunities for collecting measurements of business performance. Development understands the performance needs of the enterprise more fully and produces systems which comply with the performance requirements specified for them. Development's performance agenda has started to align with those of the enterprise. The measurements captured for each system in production are not only consistent with each other, but they are starting to support consistent cost reductions.

At levels 4 and 5, an increasing percentage of enterprise goals are visible to all. All divisions are communicating effectively about ways to advance the goals of the enterprise. The effects of a change made early or late in the system life cycle, from business management to development to operations, are discussed, rationalized, and agreed to, all in the context of enterprise goals. The measurements collected, reported, and analyzed include the full life cycle, from requirements analysis to production, and encompass system scopes from disk drive to balance sheet. The set of measurements collected and reported in each case are rationalized for ROI, as well. That is, measurements are effective and relevant from a process management standpoint—in other words, they don't ask a question without knowing what will be done with the answer.

4.5 Coupling to Business Goals

As an enterprise moves toward higher levels of maturity, performance work is better and better coupled to business goals.

At level 1, the business relevance of what little performance work is done is minimal to none.

At level 2, coupling or relevance is still limited.

At level 3, the coupling of performance work to business goals is good, but only as far as the quantitative requirements given to developers go.

At level 4, the coupling is tight. Business goals are integral to performance requirements.

At level 5, by definition, the coupling is constantly improving; quantitative data collection and performance goals are in synergy at all levels of systems from top to bottom.

4.6 Training

Performance training of someone thrust into a level 1 situation is likely minimal. The performance worker may be a system administrator or a developer forced to deal with urgent performance issues.

The performance worker at level 2 represents greater specialization, but that specialization may not be backed up by specific performance training. Much of the performance expertise applied here may have been acquired through the "school of hard knocks."

The level 3 performance specialist needs training in one or more specialty areas, suiting the system's logical and physical architecture: software performance engineering (SPE), discrete simulation models and/or queuing network models, mainframe workload management systems, operating systems internals, benchmark driver software, and related areas. Architects and lead developers in a level 3 process need at least SPE training tailored to developer roles. Both types of workers understand basic descriptive statistics at a minimum (mean, standard deviation, and so on) and can apply them to their work.

The level 4 performance specialist needs not only specific performance training, but broad exposure to at least some management theory and practical experience in a variety of enterprise situations.

At level 5, management science dominates. Leaders at level 5 need experience in performance disciplines, management consulting, and a wide range of related enterprise-level skills. The most important qualification for workers in a level 5 situation, though, is the willingness and ability to understand the big picture, to understand how decisions made in one division affect others, and to keep communications channels open.

4.7 Documentation

Level 1 sees little or no documentation produced.

At level 2, alerting thresholds and processes are documented and shared between operational support teams.

At level 3, a wide variety of supporting document types are generated and evolved, including these:

- Overall performance plan
- Response time budget
- Derived requirements needed to fulfill and/or verify compliance with performance requirements
- Benchmark and/or volume test plans
- Monitoring plans and processes
- Performance/capacity baseline

At level 4, the emphasis is on process- and business-level documents. Besides system performance requirements, the requirements statement also must address the business performance measurements to be captured and reported, as well as how those measurements will be fed back into business improvement. For example, who will receive and analyze business performance measurements and what form (format) will they take? What determines when sufficient evaluation and action have taken place? Those business performance measurements drive new requirements given to development.

At level 5, further documentation is exclusively at the process level, specifying how divisions interact and cooperate, how changes are introduced, evaluated, and deployed, and where the points of control and coordination are.

4.8 Scope And Volume Of Statistics

At level 1, the statistics captured and reported are of limited scope and volume.

At level 2, the performance worker collects data on dozens or hundreds of servers.

At level 3, the performance worker may collect smaller volumes of data, but reflective of wider scope, earlier in life cycle processes (such as volume tests on a system under development).

At level 4, the enterprise captures larger volumes of data, with the scope focused on business measurements (user productivity, etc.) and harmony of application systems across the enterprise.

At level 5, the scope of data collected and reported includes process statistics for a wide range of activities, groups, and levels of systems. The data volume is thus much more variable.

5 How Do We Make Progress In Performance Process Maturity?

The answer to this question is much tougher than simply describing maturity levels and their benefits. Nevertheless, here are some suggestions; their usability obviously depends on the skill sets of those wishing to effect change and improvement, as well as on the resources available (people, software, time, etc.).

It is important to note that the activities and remedies proposed here are not in a strict, inflexible order. They are only a starting point; readers will likely have to adapt them to their own, unique situations.

5.1 From Level 1 to Level 2

This analysis assumes that the applications involved are both high scale (lots of users) and high concentration (high percentage of time that users devote to using the applications). Or, perhaps, the cost of poor responsiveness or downtime is very high for reasons unique to the situation. To the extent that an application or group of applications vary from those ideals, the ROI of improvement efforts will also vary.

For each incident observed:

- Document the cost of the incident, including the number of people impacted. Be sure to tally users and support team members.
- Analyze the life cycle process failure which produced the incident—not to assess blame, but to understand how to make things better, independent of personalities.

From the information you have collected, try to develop a thorough picture of how things have gone astray and how much it is costing for things to remain as they are. Keep your management informed of these issues, but using a matter-of-fact style.

At the same time, for each application in production:

- Document the types of monitoring available. Monitoring options available may include resource measurements, application-specific response times, internal response time measurements produced by systems software, or a combination thereof. It is important to separate the applications themselves (or at least sets of them) from the machines they run on, for a couple of reasons:
  - First, it will almost always be simpler to tie system incidents and lost user productivity to applications because, especially in a large organization, that is how the records are most likely kept.
  - Second, the machine complement is likely to evolve over time. Server power increases with each new clock cycle bump, potentially also increasing the financial incentive to consolidate stable applications onto fewer servers. Fewer servers may mean a different monitoring strategy is needed.
- Document the approximate number of users affected by each application, which also implies the cost of downtime or poor performance.

These facts will help one rationalize the monitoring approach to take and which efforts will produce the most improvement and cost savings. Automation should be considered a top priority, because it will allow the performance worker to take the highest-level view possible. Summary statistics over a machine population and trend identification, requiring lots of automation, are critical to level 2 because applications have not yet been tuned for optimal performance.

Response time measurements are key to achieving order. Why? They give the clearest view of the user experience. If the user experience is positive, there is rarely a need to dig deeply into resource measurements. If users are suffering, response time measurements not only tell how badly they are suffering, but also reveal when corrective actions have produced benefits. They also confirm or deny anecdotal reports of poor system responsiveness, allowing efforts to be focused on the most critical problems.

5.2 From Level 2 to Level 3

The guidelines for getting from level 2 to level 3 are much like those for getting from level 1 to level 2. They include documenting costs and analyzing failures, as before. But because improvement from level 2 to level 3 involves multiple teams, direct action by a single performance worker is limited. The worker has to communicate findings of lost productivity and ineffective process through the management chain. For this reason, a firm grasp of facts, figures, and the way existing processes work is essential. Quality of documentation and presentation is the key to effectiveness; see “Presenting to Non-Technical Managers” [SWIS01]. Once those facts and figures are presented, the performance worker is pretty much dependent on the management chain to change broken processes.

By the same token, the way that one fixes what's broken depends on what is broken:

- **Development receives poorly defined or no performance requirements at all.** Operational management needs to communicate the costs to the business management function and to development. The business management function needs to be educated on the costs incurred by poorly stated or nonexistent performance requirements. Business management needs to create well-defined performance requirements that reflect the needs of the enterprise through the whole life cycle and across all divisions involved. Most importantly, business management needs to supply development funding consistent with performance and operational needs, rationalized for ROI. And of course, development needs to be made aware of how to produce systems that meet well-defined performance requirements.

- **Development receives performance requirements, but is ineffective in applying them,** producing systems with poor or unreliable responsiveness or throughput. The cost of poor performance in production needs to be documented and communicated to two entities: the business management function that funded the development and to the development management that produced the system. The operational management that suffered the cost is the most likely party to take the lead in the communication and resolution process.

- **Operations mismanages hardware acquisition, configuration, deployment or support.** The software given to operations may be capable of meeting performance goals, but is deployed or supported in such a way that performance or user productivity suffers. Insufficient or incorrect hardware is bought, or the hardware or software is configured incorrectly. Again, all three major participants must work together to resolve this failing, as well.

One can doubtless imagine other ways performance failures can occur, but most will fall into one of the three categories described above.

5.3 From Level 3 to Level 4

Maturity level 3 assumes that business management, development, and operations are in alignment with respect to performance goals. Communications channels are open in all directions. The next steps are:

- Business management identifies which user productivity and other business-level measurements should be captured in application logic. A performance worker with a clear understanding of the application and how it affects business measures may need to communicate the opportunity. High-scale, high-concentration applications may offer the biggest payoff. The steps of analysis should approximate these, with examples in parentheses:

    - Identify **relevant business goals** (maximize sales).

    - Identify **business performance factors** which support those business goals (increase user productivity).

    - Identify **specific performance metrics** which comprise or support those business performance factors (calls per hour or call duration). In the ideal case, these performance metrics can be measured from within the application.

    - Identify **components of those performance metrics** which are amenable to measurement within the application (time spent per application screen, sequence of screens accessed).

- Business management, development, and operations participants together develop an ROI for an initiative (or additional requirements for a new system). That initiative should address the selected metrics. Costs which must be accounted for include these: requirements development, development of the code which collects and saves measurements, additional hardware resources for data collection, additional network loading for monitoring traffic, development of the collection and analysis mechanism (or deployment of COTS software addressing that function), and a worker to analyze the data and produce reports and recommendations.

- Executives who provide funding evaluate the ROI and overall business case. If the ROI

(ratio of projected benefits to estimated costs incurred) is high enough, the effort can move forward.

Of course, the process described is not fundamentally different from the way any proposed applications functionality requirement is developed and rationalized. The difference is that business performance requirements (such as those which monitor user productivity) look inward at the business. Rather than chasing new business, business performance requirements help to optimize the business in place or ensure that a new initiative will generate the maximum possible ROI.

Evolution of an enterprise architecture is an even tougher problem—perhaps even tough enough to justify a separate maturity level. Locating the touch-points to optimize individual applications or related application groups and their business environments may not be difficult. By contrast, though, changing the behavior of a large variety of user and support groups, as well as the corresponding development groups, requires access to a huge array of cost and benefit data. Without access to that data, even a committed group with their executive's support may experience difficulty. Still, for groups with access to the right data and an understanding of overlapping functions and conflicting agendas, the way may be clear.

5.4 From Level 4 to Level 5

Initiative to take an enterprise from level 4 to level 5 is mostly owned by the executive suite. Process improvements which cross major divisions require a vision of perfection of the enterprise available only to those in a position to implement that vision. That vision must encompass a process culture which gives visibility to enterprise goals and consistently and fearlessly adjusts processes to address those goals. Workers and managers across all divisions report, support, and capitalize on opportunities to optimize business processes.

6 Comparison With Software Process Improvement Models

Software process improvement (CMM-style) has little or nothing to say about the achievement of management goals outside the development arena. Yet, a comprehensive and mature overall management process must balance the qualitative and functional orientation of mature development processes with an appreciation and leveraging of the quantitative aspect of system performance—how fast, how many, and at what cost, and do so at all levels.

This maturity model was inspired by the SEI's CMM, but there are important differences:

- **Skipping levels.** CMM specifically disallows skipping levels. The maturity model specified here allows skipping levels. To be sure, characteristics of level progressions in both models are derived from priorities—which things need to be done first.

- **New organizations starting above level 1.** In theory at least, a new organization can start above performance maturity level 1. In fact, the time to build foundations for level 3 and above is from the first, while expectations are still being formed.

There is much in common with CMM, though:

- **Process emphasis.** Performance work necessarily involves many people and the way they cooperate. A single performance worker, acting alone, cannot effect sweeping change.

- **Change takes time.** Because groups in a mature enterprise with a variety of charters and agendas each own a piece of the puzzle, it can take years to align their agendas and refine processes to achieve comprehensively better performance.

7 Suggested Next Steps and Conclusions

7.1 Where Do We Go From Here?

The proposed performance process maturity model could itself be taken to the next stage by incorporating an exhaustive specification of performance activities and process steps that must be performed according to system attributes and classifications in various dimensions. (For a rough example, an online transaction processing system requires somewhat different performance activities from a batch-oriented application.) CMM could serve as an example of such an exhaustive specification. The depth and refinement of activities specified for each maturity level would still conform to the present model. Such a thoroughgoing model could help turn the art of performance analysis into a science, at least in its application.

In a similar vein, more work is needed to perfect a performance science of layers of systems. That is, as soon as one advances a draft architecture of a computer *or* business system, we need to be able to comprehensively and unerringly identify:

- The metrics associated with it at all levels of scope
- What arrival rates, response times, and costs unfold from outermost system to innermost
- The costs and risks associated with failure to identify all these factors

The aim is to predict how the system will behave in production, with increasing levels of precision.

A key element of that science could be a detailed taxonomy of performance defects. There exists at least one taxonomy of software defects [BEIZ90] which provides a guide to the thought process needed to create such a taxonomy. From a systems-within-systems perspective, though, a definitive taxonomy of performance defects needs a top-level classification describing the level of scope considered.

The proposed model presents what amounts to a checklist by which one can assess one's organization. Yet this model addresses only the structural or qualitative aspect of an enterprise. Further work is needed to develop this model's quantitative counterpart, assessing how much performance activities cost versus the benefits they bring. In other words, given a type of system, a number of users, and other parameters, what performance activities are needed? How much do they cost over the entire life cycle? What costs and risks are likely if they are not incorporated into the life cycle?

7.2 Conclusions

While much of the background of this paper comes from a limited number of enterprises, the author believes it nonetheless sketches a picture of maturity which will apply to a variety of enterprises and systems. The questions it asks are fundamental:

- How are we ensuring the performance of systems?
- Are we presenting the right requirements to developers to ensure performance in the real world?
- How does performance work dovetail with the needs of the enterprise, the larger system?

The last question is most important, because it embodies the fundamental assumption of performance work. Saving time and saving system resources at all levels of scope saves money.

This paper has proposed a model of performance process maturity, beginning from a state of low discipline, leadership, and order, and progressing through increasing refinement, rationalization, and common sense. For each level of maturity, it has described typical patterns of behavior, likely areas of strength and weakness, and ways to move ahead.

This paper presents ideals for increasing performance process maturity. It is hoped that by presenting those ideals in the context of a structured series of steps, it helps inspire both performance workers and managers to reach for those ideals.

References

[BEIZ90] Beizer, B., *Software Testing Techniques, Second Edition*, Van Nostrand Reinhold, ISBN 0-442-20672-0 (1990).

[SMIT02] Smith, C. U., notes from Software Performance Engineering seminar held April 28 through May 2, 2002, in Santa Fe, New Mexico, page 1-4.

[SWIS01] Swisher-Roth, A., "Presenting to Non-Technical Managers", from the Computer Measurement Group (CMG) 2001 Conference Proceedings, paper number 300.

Trademarks