Preface

With the fast development of networking and software technologies, information processing infrastructure and applications have been growing at an impressive rate in both size and complexity, to such a degree that the design and development of high performance and scalable data processing systems and networks have become an ever-challenging issue. As a result, the use of performance modeling and measurement techniques as a critical step in design and development has become a common practice. Research and development on methodology and tools of performance modeling and performance engineering have gained further importance in order to improve the performance and scalability of these systems.

Since the seminal work of A. K. Erlang almost a century ago on the modeling of telephone traffic, performance modeling and measurement have grown into a discipline and have been evolving both in their methodologies and in the areas in which they are applied. It is noteworthy that various mathematical techniques were brought into this field, including in particular probability theory, stochastic processes, statistics, complex analysis, stochastic calculus, stochastic comparison, optimization, control theory, machine learning and information theory. The application areas extended from telephone networks to Internet and Web applications, from computer systems to computer software, from manufacturing systems to supply chain, from call centers to workforce management.

The aim of this book is to bring together the latest advances in methodology and techniques of performance modeling and engineering, ranging from theoretical advances to system and architecture developments, from technology to economics, from academic innovations to engineering processes, from statistical analysis to system control, and from enterprise systems to computer networks. The purpose for this collection is to promote innovative research in these emerging topics, to bridge the gap between theory and practice, and to stimulate the use of these new developments.

The book is organized in two parts. Part I focuses on performance design and engineering, introducing new methodologies and considerations including machine learning, network economics, online advertising and performance engineering. Part II concentrates on scheduling and control, covering new developments in Internet traffic routing, network scheduling, and modeling and control of computer systems. Each chapter is self-contained, including both a broad survey of the topic and the technical challenges and solutions. Below we briefly summarize the content of each part.

Part I on performance design and engineering comprises four chapters. Alina Beygelzimer, John Langford and Bianca Zadrozny present in Chapter 1 an overview of machine learning methods that can be applied to performance prediction. Specifically they focus on reduction techniques that transform practical problems into well-studied machine learning problems, which can then be solved using existing base-learning algorithms.

In Chapter 2, Dave Jewell presents an introduction to Performance Engineering and Management Method (PEMM), a process and a methodology that integrates many useful performance modeling and engineering techniques and provides a holistic approach to addressing the risks related to information technology performance, capacity and scalability.

Jean Walrand discusses economic models of communication networks in Chapter 3. By exploring the interaction of network designs and economics and by discussing a number of concrete models of network neutrality, service differentiation, and security, he illustrates the importance of the economic impact on design choices.

In Chapter 4, Jon Feldman and S. Muthukrishnan provide a comprehensive overview of the auction system for sponsored search advertising. They show examples of research in algorithmic, optimization and game-theoretic issues in bidding, pricing, and market design of such systems. They further describe the mathematical and algorithmic issues and challenges in sponsored search advertising.

Part II on scheduling and control comprises three chapters. In Chapter 5, M. Kodialam, T.V. Lakshman and Sudipta Sengupta provide a survey on recent developments in oblivious routing of Internet traffic. They describe two broadly used oblivious routing schemes and discuss the deployment, performance guarantee, and various routing architecture and protocol issues in today's Internet.

Devavrat Shah presents in Chapter 6 a survey on some of the latest results on design methods and analysis techniques of distributed scheduling algorithms in networks. He argues that a canonical way to design network algorithms is through a message-passing paradigm, through which the various approaches such as randomization, belief propagation, heavy traffic theory, flow-level modeling can be unified.

In Chapter 7, Tarek Abdelzaher, Yixin Diao, Joseph L. Hellerstein, Chenyang Lu and Xiaoyun Zhu provide an introduction to control theory for computing practitioners, and illustrate a number of recent successes in applying control theory to managing applications in the areas of database systems, real-time systems, virtualized servers and power management.

In summary, this book provides new perspectives of performance modeling and engineering. It allows the reader to understand both theoretical and practical issues, and to develop both mathematical frameworks and engineering processes. It also establishes a connection between market mechanisms and the Internet and Internet applications. It provides new insights in the application of control theory and optimization techniques on computer systems and computer networks.

This book is used as the textbook for the ACM SIGMETRICS 2008 Tutorial program. We believe that the book can provide the larger SIGMETRICS community and networking and systems communities theories and techniques that can help improve the performance of computer systems and networks. The book will introduce

practitioners to new methodologies and techniques and pose practical problems to researchers from both the theoretical and applied communities.

We sincerely thank all authors and tutorial presenters for their contributions to this book and to the ACM SIGMETRICS 2008 Tutorial program. We also would like to thank our colleagues, Fred Douglis and Srinivasan Parthasarathy for their timely assistance.in reviewing the book chapters. Without all of the work and support from both groups of people, this book would not have been possible. Last but not least, we are grateful to ACM and IBM Research for sponsoring the tutorial program of the SIGMETRICS 2008 Conference.

Zhen Liu
Cathy H. Xia

Chapter 2
Performance Engineering and Management Method — A Holistic Approach to Performance Engineering

Dave Jewell

Abstract Experience has shown that there is no one "silver bullet" for achieving acceptable performance in IT solutions. Early performance models help us ask the right questions but may not be as accurate as we would like in predicting future performance and capacity utilization. Performance testing of the solution once it is built gives us more accurate information, but may occur too late in the life cycle to permit fixing persistent performance problems in a timely manner. The Performance Engineering and Management Method (PEMM), first proposed by IBM in 1998, integrates these and other techniques into the Information Technology (IT) solution development life cycle, yielding a more comprehensive approach to addressing the risks related to IT performance, capacity and scalability. This paper provides an overview of the major themes of PEMM, including examples of its application and potential synergy to be gained by combining PEMM with other disciplines such as Information Technology Infrastructure Library (ITIL®) Capacity Management.

2.1 Background

The Performance Engineering and Management Method (PEMM) was first formally outlined in March 1998, in the form of a reference document for IBM system architects. PEMM was based on the experience of practitioners in IBM's United Kingdom organization. Many of the life cycle principles embodied in PEMM had already been used successfully during the 1980s, with associated technical papers presented at forums such as IBM's Information Management System (IMS) Guide and Computer Measurement Group (CMG), and had been included in client education between 1988 and 1994.

Dave Jewell
Performance Engineering Practice, Systems Engineering, Architecture & Test (SEA&T), IBM Global Business Services, e-mail: jewell@us.ibm.com

The developers of PEMM had come to believe that the best way to address system performance was to proactively manage it throughout the solution development life cycle, from the system's inception through its deployment and maintenance in a production setting. It has been IBM's experience that throughout the last twenty five years; the principles within the PEMM have been long lasting and continue to deliver high value.

In an information technology (IT) context, *performance* generally refers to the speed with which a system[1] accomplishes the tasks it was designed to do, and is commonly expressed using measurements such as response time and throughput. Performance is closely related to and often dependent upon *capacity* (the measured ability of a system to perform work) and *scalability* (the ability of a system to accommodate workload growth). The dilemma faced by the IT industry with respect to performance and capacity is at least two-fold:

1. While systems, the technology on which they are based and the demands placed upon them are leading to increased complexity and therefore greater risk of poor performance, software engineering as a whole has traditionally been focused on ensuring the *functional correctness* of those systems rather than addressing performance, capacity and scalability concerns.
2. Even when so-called best practices for performance are followed during a system's development, it is difficult to grasp the full extent of inherent performance issues until all the components of the system can be integrated and tested to see how they perform together.

This has led to the situation where the industry has been working backwards over time to get a better handle on system performance. Poor performance in production led to the recognition of the value performance testing prior to deployment to identify and address performance issues. This in turn has led to focus on performance best practices, the notion of designing for performance, estimating and modeling in attempt to predict future performance, and creating requirements which unambiguously describe the users performance requirements and clearly link them to the needs of the business. The intent of the PEMM is to organize these activities in a cohesive manner, and to successfully manage performance requirements and mitigate risks throughout the software development life cycle.

[1] Throughout this paper, *system* refers to an interrelated group of IT components working together; *solution* refers to the application of IT systems, products, etc., to solve a customers business problem and/or address a customers business opportunity.

2.2 What is Performance Engineering?

Performance Engineering (PE) is a technical discipline which aims to ensure that a development project results in the delivery of a system which meets a prespecified set of performance objectives [2]. This is done by:

- managing the performance risk of a project,
- controlling or coordinating activities in the project that have an impact on performance, and
- applying specialized performance estimation and design skills to the architecture of the system under development.

In order to properly apply PEMM, it is important to understand the reasoning behind this view of performance engineering.

- **Performance objectives should be specified prior to development of the system.** A system that must consistently furnish subsecond response time or support 10,000 concurrent online users may be more expensive and time-consuming to design, build and support than an identical system with less stringent performance objectives. Requirements related to scalability, performance, capacity and scalability should be given the same level of attention given to the system's functional requirements when considering architecture, design and implementation alternatives.
- **Performance risks must be managed rather than eliminated.** It is generally not feasible to eliminate *all* risk of poor performance of a future system. The best we can hope to do is to find a cost-effective balance between performance *risk containment* (i.e. PE) activities and performance *risk acceptance*.
- **Achieving good performance requires coordinated effort.** Since so many groups and individuals are involved in defining, designing, implementing and deploying systems, typically there will not be just one group that makes or breaks performance. Coordinating PE activities throughout the life cycle will help to achieve the best possible results.
- **performance engineering is more than just waiting for performance testing to start.** There are a number of proactive PE techniques which can be applied even before performance testing starts in order to reduce performance risk. Proactive techniques such as performance modeling, performance budgeting, and application profiling can be used to get an idea of what performance will be like before an integrated system is available for performance testing purposes.

[2] The term Software Performance Engineering (SPE) is sometimes used in some of the literature pertaining Performance Engineering (PE) [10, 11, 12], although PE as discussed in this paper refers to information technology solutions in general and not just software.

2.3 Overview of PEMM

The central notion of PEMM is that performance engineering activities must be linked to the relevant portions of the system development life cycle. Figure 2.1 shows the conceptual view of PEMM in the context of a "typical" IT solution development project.

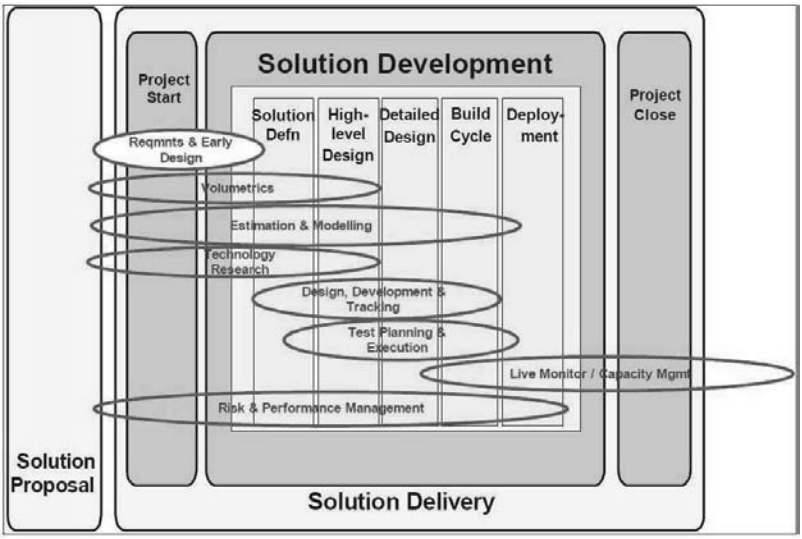


Fig. 2.1 Conceptual View of the Performance Engineering and Management Method (PEMM)

PEMM consists of eight interrelated, overarching "themes" which are active at various stages of the development project. The vertical bars depict stages which could be expected in a typical development project (proposal, project start, definition/requirements, design, build/test, deployment, project close). While the solution development process itself can take place with or without the use of PEMM, the ovals in the diagram above illustrate the relative chronology of the PEMM themes with respect to the solution development stages.

The PEMM themes can be described as follows:

- **Requirements and Early Design** - Performance activities occurring at the start of the development life cycle;
- **Volumetrics** - Quantitative elements of the workload and its data are vital to establishing the basis for the performance of the system;
- **Technology Research** - Gaining understanding of new solution elements (hardware, software, network, etc.) for which limited performance information is available;
- **Estimation and Modeling** - The central core of the predictive work done in Performance Engineering;

- **Design, Development and Tracking** - Helping the designers and developers deliver on the performance objectives;
- **Test Planning and Execution** - Testing to validate that the system will meet its performance and capacity objectives;
- **Risk Management / Performance Management** - Assessing and managing of the performance risks throughout;
- **Live Monitoring and Capacity Management** - Managing system performance and capacity for the rest of its operational life.

Each of these themes will be discussed in more detail in the chapters to follow. By applying the PEMM themes in a holistic manner throughout the development life cycle, the potential exists to

- improve system performance,
- reduce or more effectively manage system capacity and related costs,
- reduce or more effectively manage project and business risks related to performance,
- reduce the cost of addressing performance defects or issues by finding them and fixing them earlier.

2.4 PEMM Theme – Requirements and Early Design

Figure 2.2 illustrates some of the key considerations associated with the "Requirements and Early Design" theme.

Requirements and early design activities may require a certain amount of negotiation and iteration to come to completion. IT architects work with business analysts to document solution requirements. These requirements not only describe the system's desired capabilities, but also the dependencies and constraints (technical, financial, etc.) being imposed on the system. The architecture team will then consider one or more candidate solutions and evaluate whether they can meet requirements. Requirements that cannot be met may be renegotiated before going forward. This process is important not only for establishing a strong *technical baseline* [3] for change control purposes, but also for setting realistic expectations on the part of stakeholders.

While all of this is true for the general case of requirements and early design, there are additional considerations that are worth mentioning in regard to performance.

[3] Baseline documents include requirements documents, design documents and other materials defining the current state of the solution under development that have been reviewed and accepted by the projects business and technical stakeholders. Changes to baseline documents are carefully managed to prevent uncontrolled impacts to the project costs, quality and schedule.

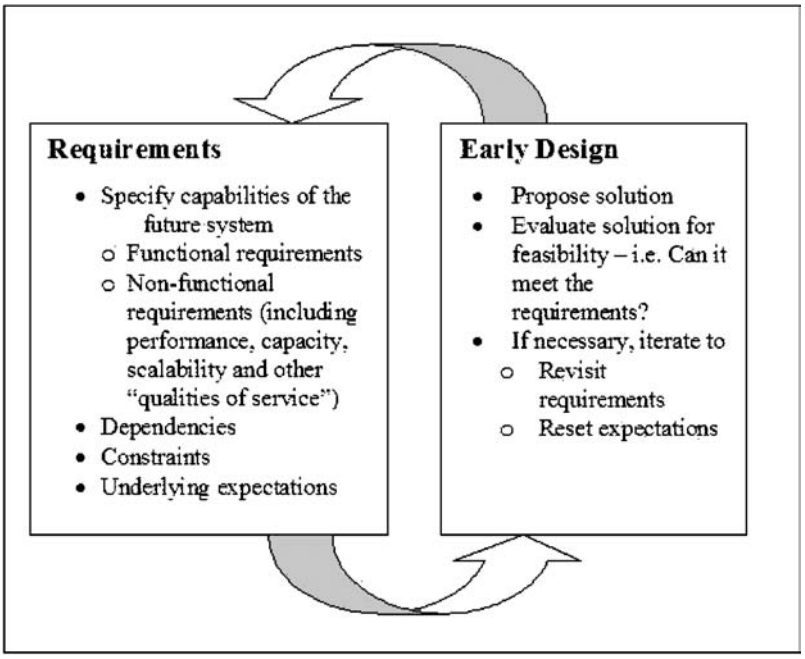


Fig. 2.2 Requirements and Early Design theme considerations.

2.4.1 Requirements and Performance

In engineering circles both inside and outside of IT, it is widely accepted that functional requirements must be understood before design can proceed – in other words, "what" is to be built must be known before "how" to build is addressed. Whether starting with formal written requirements using a waterfall development methodology or elucidating requirements incrementally using incremental development techniques, it is well recognized that systems development must be driven by requirements.

The notion of requirements can also be extended to what are referred to as *nonfunctional requirements*, i.e. assertions about the operational quality of service (QoS) constraints and dependencies which extend beyond application functionality. Menasce et al. [5] lists some typical IT quality of service characteristics which may be included in nonfunctional requirements:

- Response time
- Throughput
- Availability
- Reliability
- Security
- Scalability
- Extensibility

Architects and designers of IT systems attempt to choose the technical solution alternative which best satisfies all of the requirements, which in some cases may mean making trade-offs between requirements that may lead in different directions. For example, certain high availability measures may be more resource-intensive and impact response time or throughput. Behind some solution requirements may be contractual commitments, unstated expectations and other factors. An important first step towards achieving good performance is to ensure that performance requirements are sufficiently well documented and defined for each stage of the development process, and that they are clearly tied to the actual needs of the business. Vague or unrealistic performance requirements will hamper efforts to make the right technical decisions when trade-offs relating to performance must be made.

An important notion here is that in development projects, even the requirements themselves have their own engineering process associated with them. Requirements are to be written so that they are "SMART" (i.e. Specific, Measurable, Achievable, Realistic, Testable), as the architecture, design and build process will eventually decompose the high-level requirements into more granular components (system requirements, performance objectives, performance budgets – see also [9]). Throughout the development process, *requirements traceability* should be maintained so that all subsequent activities are clearly linked to the original requirements. While this is often done for functional requirements, this is sometimes neglected for performance and other nonfunctional requirement areas. PEMM helps ensure that nonfunctional requirements are addressed.

2.4.2 Early Design and Performance

The "Design for Performance" philosophy asserts that system performance cannot necessarily be taken for granted; therefore, when a certain level of performance is required, part of the design effort should be concerned with how the performance requirement will be met. While exact predictions of performance may be impossible to make during the architecture and high-level design stages of a project, to the extent possible, studies of the feasibility of the solution with respect to performance and other factors should be conducted.

Performance or capacity estimates made as part of an early feasibility study may be made with spreadsheets, formal sizing tools or comparisons to previously built systems. While they may not have the accuracy that future estimates would have when more is known about the solution, feasibility estimates provide a means of managing performance risk by evaluating design alternatives in terms of their effect on performance. If an area of performance risk can be flagged early, the requirements and business case can be revisited, and other technical solutions can be considered before going forward. While not all performance risk can be removed this way, feasibility estimates can disclose areas of performance risks that can be revisited in later stages of the development process.

2.5 PEMM Theme – Volumetrics

Figure 2.3 illustrates some of the key considerations associated with the "Volumetrics" theme [4]. As stated before, volumetrics refers to the quantitative elements of the workload and its data that are relevant to the performance and capacity of the system being built. Volumetrics can be considered as being in two categories: Business volumes and Technical volumes.

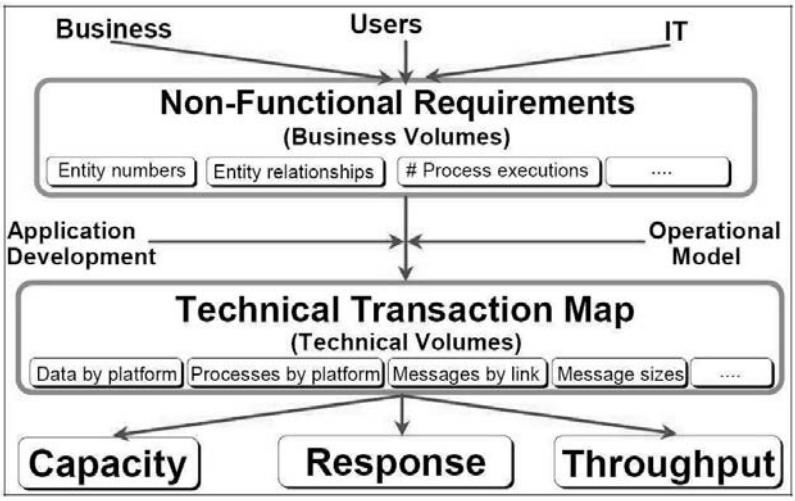


Fig. 2.3 Volumetrics theme considerations.

2.5.1 Business Volumes

Business volumes provide a business view of the quantitative elements that are important for planning system performance and capacity. These may include projections or assumptions for things such as the number of orders per day or week, the number of customers in the database and so on. At this level, the volumes are stated in terms that are relevant to the business, and may require someone with access to authoritative projections for business growth, etc., to be rendered accurately. Busi-

[4] Figure 3 includes a reference to an IBM architectural deliverable known as an Operational Model. The Operational Model (OM) defines and documents the distribution of an IT system's components onto geographically distributed nodes, together with the connections necessary to support the required component interactions, in order to achieve the IT system's functional and nonfunctional requirements within the constraints of technology, skills and budget.

ness, user and IT personnel may all be involved in developing the business volumes portion of the nonfunctional requirements.

Business volumes are often included with the nonfunctional requirements because they provide a critical context to the architects and designers of the system. For example, two systems with identical functionality may both have a mean response time requirement of 5 seconds, but if one of the systems only needs to support 300 concurrent online users while the other must support 30,000 concurrent online users, the design and capacity attributes of those two systems may need to be substantially different from each other. Similarly, the number of a key entity such as customer affects the amount of data to be processed or searched, ultimately affecting the responsiveness and resource intensiveness of data processing operations.

When developing business volume assumptions, the requirements team should take into account factors such as:

- whether a given volume element is likely to be static (not changing) or dynamic (changing over time)
- patterns of change in a dynamic volume element
- whether peak, average or point-in-time volumes are the most meaningful way to express a given volumetric's performance requirements
- the relationship between volumes and requirements for scalability (i.e. the ability to accommodate additional workload or data over time, given assumptions relative to resource capacity growth)

Business volumes are notoriously unstable, due to a number of factors. Business projections which support the volumetrics may be at best well-informed "guesses" as to the behavior of an unpredictable business climate. Projections may be overly optimistic or inflated to justify projects that would not otherwise have a viable business case. On the other hand, they may be overly low in an effort to contain costs. For critical volumes, it may be wise to come up with a low-to-high range of estimates so that options for "scaling up" or "scaling down" the solution can be considered. Corresponding assumptions should be well documented.

2.5.2 Technical Volumes

Technical volumes provide an IT solution view of the quantitative elements that are important for planning system performance and capacity. While technical volumes are based on the business volumes, they also take into account knowledge about the structure and behavior of the proposed solution gleaned from working with system architects, designers and developers.

Technical volumes are often captured in spreadsheet form using what is sometimes referred to as a technical transaction map. As a simplified example, assume that the business volumes reveal that there will be 10,000 orders per peak hour submitted on the order processing system. If the system interaction diagram reveals that each order produces:

- 1 order submit message from the client to the web server,
- 1 order submit message from the web server to the application server,
- 3 database request messages from the application server to the database server,
- 25 message database response messages from the database server to the application server,
- 1 order confirmation message from the application server to the web server,
- 1 order confirmation message from the web server to the client,

it can then be concluded that in a peak hour, order submit activity alone will generate:

- 20,000 incoming messages to the web server,
- 260,000 incoming messages to the application server,
- 250,000 incoming database requests to the database server,
- 10,000 outgoing message from the web server to clients.

In this example, it is clear that from the standpoint of the number of messages which must be handled for order submission traffic, the greatest message processing workload will fall to the application and database servers. Depending on the needs of the project, the technical transaction map can be expanded to include the volumes of data transferred and processed, message sizes and any other information that may prove relevant to estimating or modeling future performance and capacity.

2.6 PEMM Theme – Estimation and Modeling

Figure 2.4 illustrates a typical approach to the "Estimation and Modeling" theme. Estimation and modeling techniques are used to attempt to predict the performance and capacity behaviors of potential future systems, or of existing systems using potential future scenarios. The ideal means of estimating system performance is obviously to observe and measure the performance of a live system. When this cannot be done it may be necessary to develop a performance model (an abstract representation of the performance behavior of the future system) for this purpose.

Volumetrics, which have already been discussed, obviously constitute a key input to the performance estimation process. In addition, performance estimation must take into account other factors, such as:

- **Parametric costs** - The amount of a given system resource (CPU seconds, I/O operations, etc.) consumed per transaction, message or other estimating unit of work processed
- **Resource model** ? The computing hardware and other infrastructure elements which affect overall system performance, with capacity and processing power assumptions stated
- **Queuing model** - The means of predicting wait time (i.e. how long transactions are likely to wait for busy system resources to free up)

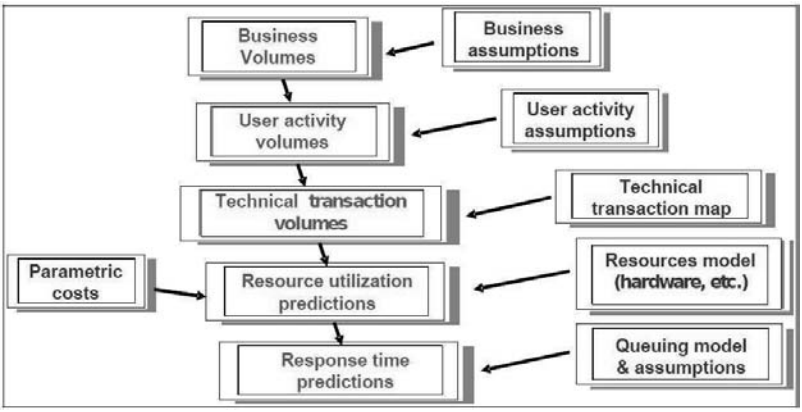


Fig. 2.4 Typical Estimation theme approach.

2.6.1 Performance Estimating Techniques

One of the more important aspects of performance engineering is the selection of the appropriate estimating technique for the situation at hand. Figure 2.5 shows some of the common estimating techniques and the corresponding cost vs. accuracy trade-offs. From a technical perspective, the difference in the techniques comes from the manner in which volumetrics, parametric costs, the resource model and the queuing model are represented and used to estimate performance.

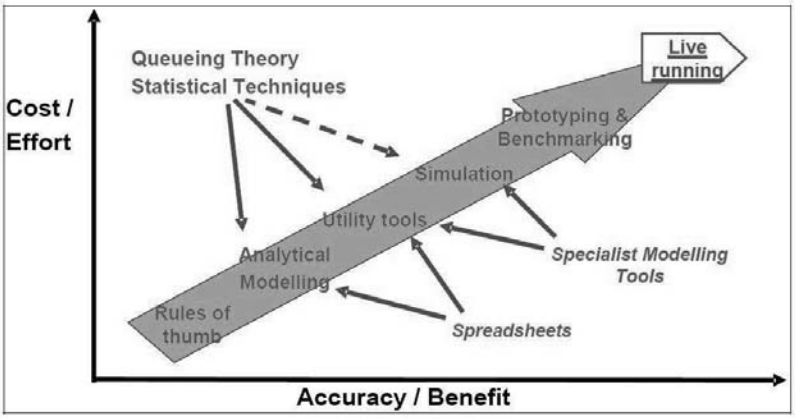


Fig. 2.5 Cost/Effort and Accuracy/Benefit trade-offs between performance estimation techniques .

Typically, more than one of these estimation techniques should be used during the life of the project. Low-cost, low-effort methods are commonly used early in the

project for feasibility study purposes, whereas the higher-accuracy methods may be employed later, when more is known about the solution design and validation of the solutions performance characteristics is deemed critical.

These estimating techniques can be summarized as follows.

- **"Rules of thumb" estimating** relies on very preliminary, simplified assumptions concerning volumetrics, parametric costs, system resources and wait times in order to deliver an estimate relatively quickly.
- **Analytical modeling** uses spreadsheets (or special purpose tools [5] in some cases) performing static calculations to make predictions of "steady state" performance and utilization of a system under a given workload. The static calculations typically take into account some amount of queuing theory and statistical techniques, in addition to volumetrics, parametric costs and system resources.
- **Utility tools** utilize data gathered from multiple performance benchmark results to facilitate the capacity sizing of systems based on reference workload types [6]. While such tools do not enable the creation of custom performance models for specific future systems, they do allow for limited specifications of volumetrics and system resources, and the parametric costs are gleaned from benchmark data tied to the reference workloads, allowing the sizing to be based on a similar workload to that of the target solution.
- **Simulation modeling** typically utilizes what is known as a discrete event simulation tool [7] to mimic the transaction processing behavior of the live system from a resource utilization and timing perspective. With these kinds of tools, the modeler must spend a considerable amount of time populating the model with the resource model, the parametric costs and transaction processing behavior (e.g. flows between the application tiers). Once this is done, the modeling tool simulates both the incoming transaction activity and the queuing/flow activity of the system being modeled, including the interaction between system components. This allows the dynamic performance behavior of the system over time to be examined.
- **Prototypes** can be thought of as partially built or preliminary versions of a contemplated invention, product or solution. The purpose of building any prototype is to learn from and leverage the experience of building or testing the prototype before making a commitment to the production version of whatever is being produced. For IT solutions, performance prototyping can serve as a means of investigating aspects of solution performance (e.g. estimating parametric costs) before a fully developed live system is available for performance testing.
- **Benchmark testing** as described here refers to the process of performance testing using a known workload before and after making a change or enhancement

[5] Accretive Technologies [1] is one vendor that offers analytical modeling tools.

[6] Examples of utility tools from IBM include the OPERA and SONOMA tools [2].

[7] A number of vendors (including HyPerformix and OpNet) and academic sources (e.g. Ptolemy II from Berkeley) offer discrete event modeling tools [3, 7, 8]. IBM Research has also developed an experimental performance modeling tool called AMBIENCE, which includes both analytical and simulation modeling capabilities. AMBIENCE also uses an inferencing engine to populate the initial model with starting parametric costs based on performance test results [4].

to the system, in order to determine whether the system's performance has been impacted by the change.

2.6.2 Selection of Performance Estimating Methods

Figure 2.5 has already introduced the general notion that increased accuracy in performance estimation comes only with increased cost and effort. We should not automatically reject the most accurate methods because of their expense or the least expensive methods because of their potential for inaccuracy. Performance estimation techniques, as is the case with most other performance engineering techniques, are intended to help us manage risks related to performance and capacity, so we must find ways to effectively balance the cost of accepting risk with the cost of containing those risks. The following recommendations are offered to help in making this selection.

1. **Consider how quickly an estimate is needed.** If an estimate is needed sooner rather than later, rough estimates and other low-effort techniques may be acceptable. This is particularly true in the early stages of a project, where the initial concern is to evaluate overall solution feasibility.
2. **Consider when the information needed for an estimate will become available.** If a certain estimating technique requires information that is not yet available, that technique may need to wait until later.
3. **Estimating techniques are not necessarily mutually exclusive.** While it may be tempting to use performance modeling in lieu of performance testing to reduce expense, the abstract nature of any performance model means that its accuracy has limitations. Rather, use performance modeling to help highlight the areas of risk that performance testing should focus on, and harvest the performance testing results as a means of calibrating the performance model for future use and achieving greater accuracy.
4. **Consider the return on investment.** Simulation modeling and performance testing often require significant investment in tools, training and development to be beneficial. Additionally, performance testing requires investment in a "production-like" hosting environment and test hardware. For these techniques in particular, there generally needs to be a return on investment in the form of reuse of the modeling and/or testing assets over time in order to justify the investments. Estimation efforts that will result in one-time, "throw-away" work may be better off using less expensive estimation methods.
5. **Consider the risk and criticality of performance for a given project.** Where poor performance has serious implication or the risk of poor performance is high, additional investment in performance estimating may be justified. Consider using performance estimation approaches which highlight likely risk areas, targeting those for additional estimation, testing and measurement.

2.7 PEMM Theme – Technology Research

A great deal of performance engineering requires dealing with uncertainty. This is certainly true of the "Technology Research" theme. Technology research is used here to describe techniques for gaining understanding of new solution elements (hardware, software, network, etc.) for which limited performance information is available. Frequently, the process of finding the data needed to populate a performance model (e.g. with volumetrics, parametric costs, etc.) ends up driving much of the technology research which takes place.

Typical approaches to technology research include the following.

- **Compare the future system with similar systems that already exist.** This is potentially one of the best sources of data available, particularly if the existing system is comparable in terms of the execution platform, volumetrics and so on.
- **Use published benchmarks to get an idea of the relative performance of solution alternatives.** Publicly available benchmarks are published by independent standards organizations such as TPC or SPEC, or in some cases by vendors, as a means of assessing solution alternatives using a common standard (e.g. to compare the relative processing power of two different server models). A benchmark usually involves a reference workload to be executed and a set of measurements to be gathered. Benchmarks may be executed either by vendors or independent test organizations that have an interest in evaluating or publicizing benchmark results for competitive evaluation purposes. The key thing to keep in mind is that benchmark results are only useful to the extent that the benchmark workload is representative of the projected system workload being modeled.
- **Consider using information from other reliable sources.** These may include published books, consulting reports, manufacturer specifications, Internet sources, or people with relevant experience.
- **Consider technical prototypes when needed information is difficult to get by other means.** As discussed earlier, performance prototyping involves building a partial solution to estimate performance attributes of the future solution.
- **Gather measurements from the system under development.** While this information will not be available at project planning time, information gleaned by development may prove very useful for refining existing performance models.

2.8 PEMM Theme – Design, Development and Tracking

Even if performance requirements have been well constructed, the solution is well architected and performance estimates show that the performance goals are attainable, ensuring that all of this comes together at implementation time is the responsibility of the development team. Performance considerations should be taken into account during detailed design and coding, and progress against performance goals should be validated as components become available for development testing.

From a process perspective, development activities are often carried out by someone other than the performance engineer or architect who helped to establish the original solution requirements, solution architecture and performance goals. Ensuring that continued progress is made towards meeting these goals requires close cooperation between all concerned. Fortunately, there are a number of approaches which can be successfully applied to address performance at implementation time.

2.8.1 Recognizing Performance Patterns and Anti-Patterns

Connie Smith and Lloyd Williams [9] described the notion of *performance principles, performance patterns* and *performance anti-patterns*. Table 2.8.1 illustrates each of these concepts in greater detail.

Concept	Description	Example
Principle for good performance	A general rule for creating designs that help achieve performance objectives	The principle of locality states that actions, functions and results should be located close to the computer resources, in order to reduce processing overhead.
Performance pattern	A frequently used design approach that tends to yield good performance results because it successfully exploits one or more principles of good performance	The alternate paths pattern helps improve the performance of data networks by providing multiple ways to send TCP/IP traffic from point A to point B, thus relieving network congestion.
Performance anti-pattern	A frequently used design approach that tends to yield poor performance results because it is inappropriately applied, or because it violates one or more principles of good performance	The one-lane bridge might be an acceptable solution for crossing a creek on a secluded country road, but would be an anti-pattern for a major highway crossing a river, or by analogy, for a low-bandwidth connection used where a high-bandwidth connection is needed.

Table 2.1 Performance Principles, Performance Patterns and Performance Anti-Patterns explained.

Smith and Williams list several principles, patterns and anti-patterns relating to performance, and quite likely more instances of each could be listed. The point is that a key to designing IT solutions that perform well is the understanding of design patterns that either do or do not work well from a performance perspective, and why that may be the case. Ultimately, performance is only one of several considera-

tions that IT architects and designers must take into account as they create solution designs. However, an understanding of performance patterns and anti-patterns will help in making the inevitable trade-offs between performance goals and other solution requirements.

2.8.2 Designing for Performance

Software engineering practitioners have long recognized the need to design for functionality, and accordingly, software quality techniques and best practices have been instantiated again and again, taking the forms of waterfall development, structured programming, iterative development, object-oriented development, agile development methods and so on. The means of designing for performance, however, have arguably been less conspicuous as part of software engineering practice.

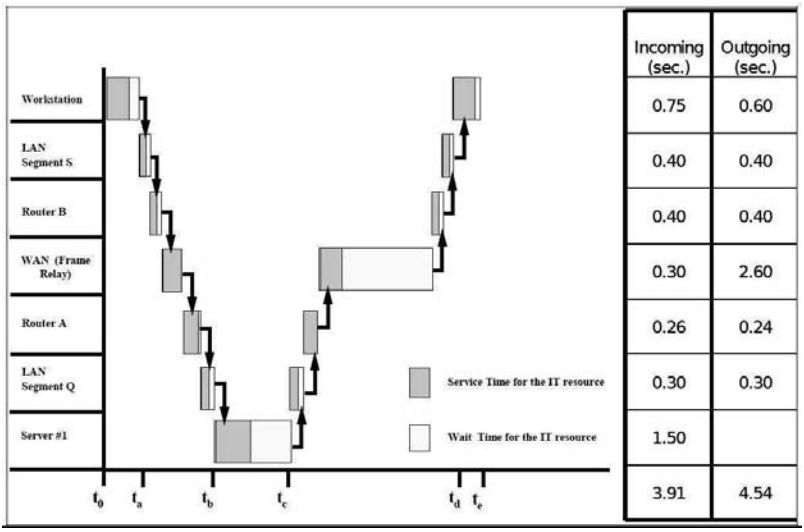
The "Design for Performance" philosophy is one that takes a goal-oriented approach to developing IT solutions that perform well, rather than taking it for granted that everything will somehow get worked out in the end. This involves more than just having programmers keep a handy checklist of Java and SQL performance "dos and don'ts". "Design for Performance" essentially means that performance goals or targets are consciously set, and design and implementation decisions are made and validated with respect to their effects on performance, in such a way that performance does not become a "nasty surprise" at the end of the project.

As much as possible, this means that design decisions creating good performance reduce the need to make "tuning" changes to the implemented solution after the fact. Applying the correct performance patterns, avoiding the correct performance anti-patterns, leveraging the advanced performance capabilities of the implementation platforms and then validating performance throughout the development process will go a long way towards making our performance engineering activities more proactive and less reactive.

2.8.3 Performance Budgeting

As with functional requirements, initial performance requirements are typically defined at a high level for business and end-user audiences. However, in order to be useful for tracking at the development level, they must eventually be decomposed into more granular components as the solution design takes shape. These more granular, component-level goals for performance are often assembled into what are known as performance budgets. A performance budget allocates time or system resources (not money!) required to complete a task for planning and estimation purposes. Figure 6 shows an example of a response time performance budget.

Initially, a response time budget such as this one might be based on estimates of the time spent processing a transaction at each layer of the infrastructure. Eventually,



Incoming (sec.)	Outgoing (sec.)
0.75	0.60
0.40	0.40
0.40	0.40
0.30	2.60
0.26	0.24
0.30	0.30
1.50	
3.91	4.54

Fig. 2.6 Example of a simple response time budget.

through the use of some kind of tracing or instrumentation, actual times for each of these components can be gathered and compared to the original budget. At that time, the process for staying in budget is similar to one that would be used on a monetary budget; developers would look for opportunities to reduce or eliminate unnecessary processing and/or make modifications to the budget that reflect the current reality. If this meant that a critical response time target was not being met after applying potential tuning changes, either the solution design or the performance requirement would need to be revisited.

2.8.4 Performance Debugging and Profiling

Developers often have available to them debugging and profiling tools that can help isolate performance problems. While the term debugging has long been associated with isolating and removing functional defects from application code, many debugging tools and techniques can help improve performance as well. For example, thread contention and memory leaks are examples of functional defects which can be found using debugging tools and techniques which also serve to improve performance.

Profiling tools and techniques, on the other hand, are generally involved with measuring application and/or subsystem performance. In particular, profiling tools and techniques are especially helpful for

- locating and resolving performance inefficiencies,
- isolating problems in a vertically large multi-tiered architecture,

- optimizing performance at the code, database or network level,
- determining whether performance budgets can be met.

2.8.5 Design, Development and Tracking Guidance

As was the case with the "Estimation and Modeling" theme, the various approaches associated with the "Design, Development and Tracking" theme should be used judiciously. Just as it is not feasible to exhaustively test an application system of any significant size, it is not generally feasible to apply all of the techniques discussed here to the entire application being developed. For example, labor-intensive techniques such as performance budgeting and application profiling are typically reserved for areas of the application which are associated with use cases for which performance is critical and performance requirements are challenging. The balance between risk and containment may dictate how each of the "Design, Development and Tracking" approaches should be used.

2.9 PEMM Theme – Test Planning and Execution

Performance testing is concerned with ensuring that nonfunctional requirements having to do with performance, capacity and scalability can be met by the solution as implemented. The job of the performance tester is made much easier if there are clear performance requirements and a proactive "Design for Performance" philosophy has been followed in architecting, designing and implementing the solution. Starting with an understanding of the designed solution and its performance requirements, the performance tester must design, implement and execute performance testing scenarios which validate the ability of the solution to meet the performance requirements.

To understand performance testing, it is helpful to understand the distinctions between functional testing and performance testing as shown in Table 2.9.

As with other kinds of testing, performance testing requires the use of effective test management practices (e.g. planning, entrance and exit criteria, designing test cases that are traceable to requirements, test execution tracking, defect management). However, the focus on performance rather than function means that different tools, skills, methods and approaches must be brought to bear. Moreover, traditional test progress metrics such as percentage of test cases complete, defect counts and defect severity may not convey the seriousness of underlying performance problems which the performance testing is starting to reveal.

Table 2.9 illustrates a number of typical activities, deliverables and work products associated with performance testing.

Performance testing of this kind is not a novel industry concept, nor is it unique to PEMM. However, with PEMM we no longer depend exclusively on performance

Functional Testing	Performance Testing
Concerned with coverage and correctness of function	Concerned with responsiveness, throughput and behavior under heavy workload with limited resources
Validates behavior in response to executing individual test cases	Validates behavior in the operational environment in response to a representative "mix" of activity
Defects note incorrect behavior.	Performance deficiencies may be a matter of degree.
Test tools and skills are oriented towards the single-user perspective.	Test tools and skills are oriented towards multi-user, system perspective.

Table 2.2 Differences between Functional Testing and Performance Testing.

testing to manage risks related to performance. Applying the preceding PEMM themes in a proactive manner can go a long way towards averting performance issues before testing or deployment. Furthermore, by applying PEMM, the performance tester can

- benefit from having performance requirements to use in developing test scenarios,
- leverage the findings of earlier estimation and modeling work to help prioritize testing efforts, and
- share test results with the modeling team to help refine the predictive models for future use.

2.10 PEMM Theme – Live Monitoring and Capacity Planning

The "Live Monitoring and Capacity Management" theme is concerned with activities needing to take place to ensure that the solution performs well in the live production environment.

The "needs of the business" should dictate the business requirements for the system, including those nonfunctional requirements related to performance, capacity and scalability. While these requirements place certain demands on the teams responsible for building the solution, they also place demands on the teams responsible for deploying, hosting and operating the solution in its production environment. For example, capacity plans must be developed and validated to ensure that the infrastructure is ready to handle the workload. Tools and procedures (preferably auto-

Test Process Stage	Activity	Deliverables and work products
Test Planning	Analyze requirements	• Nonfunctional requirements • Volumetrics
	Determine test strategy	• Selected use cases / business functions • Infrastructure scope • Application components • Monitoring tools • Test process, including phases and scenarios
	Create test plan	• Test plan
Test Preparation	Develop workload	• Workload design • Workload scripts • Workload procedures, execution plan
	Develop measurement procedures	• Measurement procedures
	Set up environment	• Prepared test environment, agreement for operations support
Test Execution	Run tests	• Test run output, measurement data • Test progress reporting
	Analyze results	• Interim results reports
	Follow-up	• Defect documentation • Problem resolution correspondence
Test Reporting	Prepare results report	• Results reports by test phase
	Publish and review results	• Review minutes
	Decision checkpoint with stakeholder	• Sign-off by phase • Deployment recommendation and decision

Table 2.3 Performance testing stages, activities, deliverables and work products.

mated) must be put in place to respond to poorly performing applications and transactions. Furthermore, there may be service level agreements between the provider (operations) and consumer (users) of IT services, in order to set expectations concerning the projected workload and resulting performance of the live production system.

2.10.1 Relating PEMM to Performance and Capacity Management

Performance management has to do with ensuring that the deployed production system performs as required, and taking corrective action when it does not. Closely related to this is *capacity management*, which has to do with planning for, deploying and monitoring physical computing resources and infrastructure to ensure that the system has sufficient resources to do its job.

Performance Management and Capacity Management of live, operational production systems are not new concepts. For years, data center operations professionals have understood that these and other systems management disciplines (problem management, change management, etc.) as being essential to running things smoothly in the production environment. In more recent years, these disciplines have become part of what is now known in the industry as *IT Service Management*, with frameworks such as the *IT Infrastructure Library (ITIL®)* beginning to codify the industry consensus on the leading practices in this area [6].

Figure 2.7 illustrates the key IT service management processes encompassed by ITIL Version 2 [8]. Note that in ITIL terminology, both performance management and capacity management are addressed by the ITIL Capacity Management process.

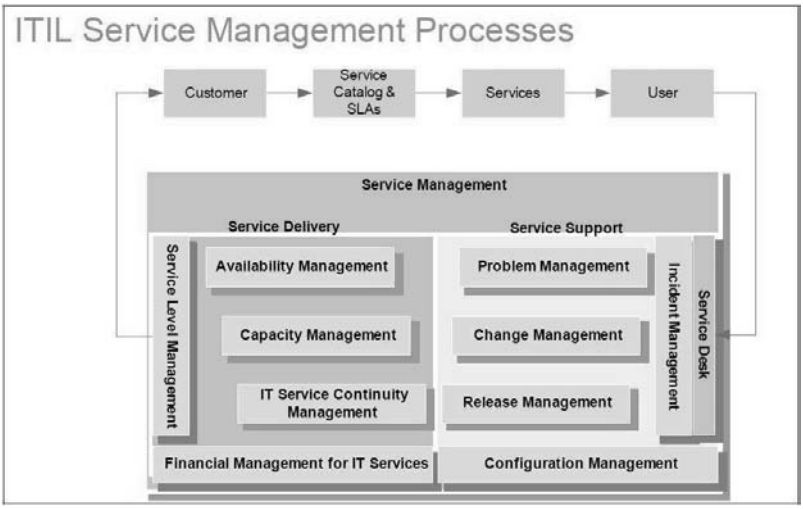


Fig. 2.7 Overview of ITIL Service Management processes.

While PEMM is not overly prescriptive about how performance and capacity management ought to be done, the chief PEMM recommendation in this area is that the potential for synergy between Performance Engineering efforts and performance / capacity management (PCM) efforts should be leveraged. This is not

[8] As of this writing, the latest version of ITIL is Version 3. However, the comments concerning the key ITIL service management processes still apply.

always as easy is it would seem to be. In many IT organizations, the differing skills, responsibilities and locations of the development and operations shops prevent the respective groups from working together, and what interfaces there are take the form of required meetings and paperwork hurdles needed to get final approvals for deployment. However, looking at the PEMM themes from a life cycle perspective, it is not hard to see potentials for achieving synergy with and improving hand-offs to the PCM team. For example:

- The "Requirements and Early Design" theme can contribute to the definition of candidate *service level requirements* (SLRs), which may become the basis for later service level agreements and for some of the development performance targets.
- The "Estimation and Modeling" theme can and should contribute to capacity planning efforts.
- The "Test Planning and Execution" theme can include exercising the performance monitoring tools and procedures that will be used in production, and providing final guidance with respect to performance and capacity expectations prior to deployment.

Achieving these kinds of synergies may become a requirement for organizations seeking to fulfill the spirit and intent of industry initiatives such as ITIL Capacity Management and the full solution life cycle scope of PEMM.

2.10.2 PEMM and ITIL Capacity Management

The ITIL service management framework has provided the industry with a widely accepted framework and vocabulary for understanding and implementing service management practices. Capacity Management is one of the major service delivery processes described in ITIL, and is defined rather broadly to address not only the capacity of IT systems and infrastructure, both also the capacity of other resources (facilities, human resources, electrical power) on which the delivery of IT services may depend. ITIL Capacity Management defines three major subprocesses:

- **Business Capacity Management**
  - Manage Service Level Agreements (SLAs) and future Service level Requirements (SLRs)
  - Study Business Plans & Create resultant Capacity Plans
  - Perform Modeling & Application Sizing for Business Services
- **Service Capacity Management**
  - Translate Business SLAs & SLRs into IT SLAs & SLRs
  - Measure Throughput & Performance for System, Network & Service
  - Perform Monitoring, Measuring, Analysis, Tuning & Demand Management

- **Resource Capacity Management**
  - Optimize & Configure Current Technology
  - Research Future Technology and other alternatives
  - Guarantee System and Service Resilience

There is a sequential flow inherent to this arrangement of the subprocesses.

1. **Business capacity** – How much business work must the solution perform?
2. **Service capacity** – How much work must the IT services do to meet the business capacity requirements?
3. **Resource capacity** – What physical resources, etc., are needed to address service capacity requirements?

Moreover, most (if not all) of the activities supporting the ITIL Capacity Management subprocesses are essentially performance engineering activities as described in PEMM. Because PEMM and ITIL Capacity Management originate from different perspectives (development vs. operational), they are actually complementary in many respects, and can be used effectively together.

2.11 PEMM Theme – Performance and Risk Management

Figure 2.8 illustrates some of the key considerations associated with the "Performance and Risk Management" theme. This theme is concerned with activities needing to effectively manage performance-related risk throughout the life of the project.

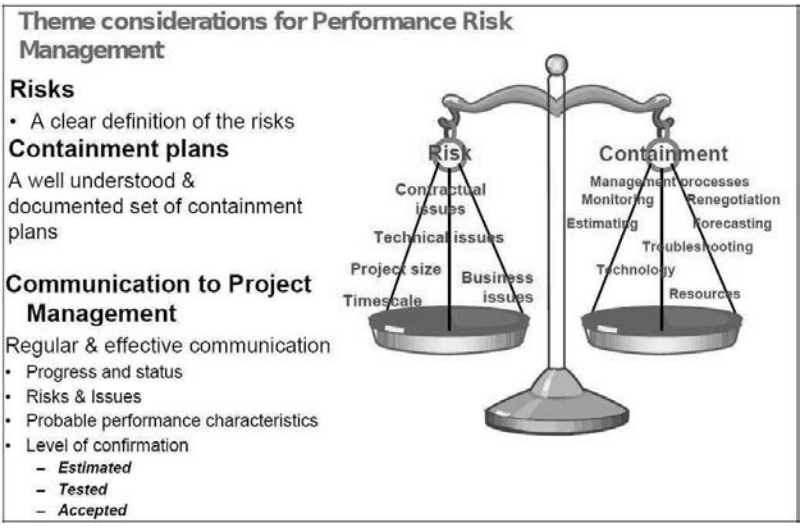


Fig. 2.8 Key considerations for the "Performance and Risk Management" theme.

Managing performance-related risks is one of the chief overall objectives of PEMM. Nearly every one of the other more "technical" PEMM themes is intended to help reduce risks related to the potential for poor performance. Just as a project manager is responsible for the management of risks to the overall project, the performance engineer must be concerned with identifying and mitigating technical risks related to performance, capacity and scalability, by engaging the right resources and/or management attention when significant performance issues or risks arise. This is done with the recognition that since it is not feasible to eliminate all sources of performance-related risks, *finding an effective balance between risk containment and risk acceptance* should be the goal.

From a project perspective, there are other things that can also be done to facilitate the management of performance risk.

2.11.1 Assignment of Dedicated Performance Engineering Resources

For complex systems with critical performance requirements, it is important to establish accountability for the management and coordination of activities needed to achieve the project's IT performance goals. The terms *performance architect* or *performance engineer* are sometimes used to describe the role of those who assume this responsibility for a given project. Depending on the project, there may be a team of performance engineers sharing this responsibility.

While the size of the performance engineering team may vary depending on the complexity and the level of performance risk associated with the project, it is not unusual to see 1% to 5% of an IT projects total cost invested in performance engineering activities, or more for complex projects [9, 10, 11, 12]. While performance engineers are typically most involved in the earlier themes of PEMM, there is a great deal to be gained if there is active PE involvement throughout the solution life cycle. The Performance Engineer may not own direct responsibility for activities such as development, testing and post-deployment performance and capacity management; however, the PE must nevertheless find ways to maintain the influence and accountability needed to help the project meet its performance, capacity, and scalability requirements.

2.11.2 Applying PEMM to IT Project Governance

The need of the performance engineer to influence activities and decisions not under his or her direct control begs the question of how PEMM relates to overall IT project governance. Most IT organizations have some level of methodology and process documentation which describes how activities such as architecture, design, development testing, deployment and live operations are to be handled, along with

a governance structure which monitors and enforces adherence to these methodologies and processes. A prudent means of applying PEMM is not necessarily to replace existing processes, but rather to ensure time and resources are allotted to integrate PE activities into the existing plans, and to identify touch points between PEMM and the local process/methodologies to ensure that adequate progress is being made towards meeting the performance goals.

2.11.3 Applying PEMM to Complex Projects

Inevitably, projects will be encountered where it is not possibly to exhaustively apply PEMM techniques to all aspects of the solution. In such cases, one must be selective about the scope and focus of performance engineering efforts. Some considerations for managing this complexity from a PEMM perspective are listed below.

- **Divide and conquer** – Rather than trying to estimate or model all the aspect of a large system or group of systems in their entirety, it may make more sense to split them into more manageable subsystems for separate consideration.
- **Prioritize** – As was stated earlier, be selective about which parts of the system warrant detailed PE studies. Concentrate on those areas where the risk and/or potential impact of performance issues is most significant. Special consideration should be given in the following cases.
- **Special cases** – The following classes of solutions often have specific performance challenges requiring special attention.
  - **Multi-channel solutions** – If a central system is being accessed by multiple channels (direct web users, telephone VRU, call center users, etc.), be aware that each channel will have different effects on and requirements pertaining to system performance.
  - **Distributed solutions** – If a common solution is replicated at multiple data centers, keep in mind that there could be site-specific performance and capacity considerations depending on how the worldwide workload is distributed.
  - **Commercial Off-The-Shelf (COTS) Packages** – It is increasingly common for customers to use vendor-provided packaged solutions or frameworks to meet their specific business needs. Ensuring that a vendor solution continues to meet performance requirements after being configured, tailored or extended to meet the customers needs is often challenging.
  - **Large programs** – Some organizations may have major conversion or transformation initiatives that take months or years to implement. These situations may have governance and technical considerations that make PE challenging in those cases.
  - **Unpredictable workloads** – The advent of the web era has shown that web-enabled workloads originating from the Internet can be difficult to predict.

Making contingency plans for additional capacity and scalability is especially important in those situations.

- **Complex interactions** – Solutions that are based on SOA and other highly modular architectures can pose workload balancing issues and other performance challenges in their own right.

2.12 Summary

The Performance Engineering and Management Method is a collection of IBM's leading practices with respect to performance engineering and performance management of IT systems. PEMM is organized as a framework consisting of eight overarching and interrelated "themes" which operate in concert with whatever locally used IT solution life cycle is in place. While the themes themselves are not necessarily unique to IBM or to PEMM, the overall framework provided by PEMM helps to conceptualize the performance engineering activities that are appropriate for each stage of the solution life cycle.

Even when users of PEMM have not had the opportunity to apply all of PEMM to a given project, having an understanding of the PEMM themes still helps to understand what things should have been done previously and should be done now to manage performance-related risks during the project life cycle. Within IBM, acquainting architects, developers, testers, and operations personnel with the concepts of PEMM have raised awareness of both the potential for applying PE in a proactive manner and the potential for achieving synergies across the themes by working together.

Implementing PEMM is not without its challenges. Managers seeking to reduce costs and shorten schedules may need convincing before they are willing to dedicate resources to performance engineering activities, and even then the scope of PE activities may be limited due to the current state of the project. Performance estimating and modeling methods are arguably immature, and commercially available performance modeling and testing tools are sometimes prohibitively expensive. Even within IBM, the process of institutionalizing PEMM has been a gradual one, relying on educational offerings, methodology enhancements, project experience and an active internal PE advocacy community to encourage PEMM adoption over time.

Nevertheless, progress continues to be made. New training has been introduced to train IBM practitioners in applying PEMM to SOA projects. PEMM concepts have been applied by IBM in both internal and commercial projects for years, and now customer demand has led to the development of commercially offered PEMM training. Moreover, the continued acceptance of process maturity frameworks such as ITIL is leading to increased awareness among IT organizations that system performance must be treated as a business imperative, and not left to chance. Given this, PEMM and similar holistic approaches to PE will fulfill a vital need in the IT industry for years to come.

Acknowledgements The author thanks Martin Jowett, Damian Towler, Chris Winter, Ann Dowling, Zhen Liu and Cathy Xia for their review of and contributions to this paper.

References

1. Accretive Technologies home page. http://www.acrtek.com/acc_home.swf.
2. E. Hung, Q. He, and J. Zhu. Sonoma: Web service for estimating capacity and performance of Service-Oriented Architecture (SOA) workloads, 2006. ftp://ftp.software.ibm.com/software/dw/wes/hipods/SONOMA_wp9Oct_final.pdf.
3. HyPerformix home page. http://www.hyperformix.com.
4. Z. Liu, L. Wynter, C. Xia, and F. Zhang. Parameter inference of queueing models for it systems using end-to-end measurements. *Performance Evaluation*, 63:36–60, 2006.
5. D. A. Menasce and Others. *Performance by Design: Computer Capacity Planning by Example*. Prentice Hall, Upper Saddle River, NY, 2004.
6. Official ITIL® web site home page. http://www.itil-officialsite.com/home/home.asp.
7. OpNet home page. http://www.opnet.com.
8. Ptolemy II home page. http://ptolemy.berkeley.edu/ptolemyII.
9. S. C. U. and L. G. Williams. *Performance Solutions: A Practical Guide to Creating Responsive, Scalable Software*. Prentice Hall, Upper Saddle River, NY, 2002.
10. SPE Experience: An Economic Analysis. http://www.perfeng.com/papers/hesselg2.pdf.
11. The Economics of SPE Panel. http://www.perfeng.com/papers/jennings.pdf.
12. What Does Software Performance Engineering Cost? http://www.cs.ucl.ac.uk/staff/ucacwxe/lectures/3C05-01-02/aswe13.pdf.

Chapter 2
Performance Engineering and Management Method — A Holistic Approach to Performance Engineering

Dave Jewell

Abstract Experience has shown that there is no one "silver bullet" for achieving acceptable performance in IT solutions. Early performance models help us ask the right questions but may not be as accurate as we would like in predicting future performance and capacity utilization. Performance testing of the solution once it is built gives us more accurate information, but may occur too late in the life cycle to permit fixing persistent performance problems in a timely manner. The Performance Engineering and Management Method (PEMM), first proposed by IBM in 1998, integrates these and other techniques into the Information Technology (IT) solution development life cycle, yielding a more comprehensive approach to addressing the risks related to IT performance, capacity and scalability. This paper provides an overview of the major themes of PEMM, including examples of its application and potential synergy to be gained by combining PEMM with other disciplines such as Information Technology Infrastructure Library (ITIL®) Capacity Management.

2.1 Background

The Performance Engineering and Management Method (PEMM) was first formally outlined in March 1998, in the form of a reference document for IBM system architects. PEMM was based on the experience of practitioners in IBM's United Kingdom organization. Many of the life cycle principles embodied in PEMM had already been used successfully during the 1980s, with associated technical papers presented at forums such as IBM's Information Management System (IMS) Guide and Computer Measurement Group (CMG), and had been included in client education between 1988 and 1994.

Dave Jewell
Performance Engineering Practice, Systems Engineering, Architecture & Test (SEA&T), IBM Global Business Services, e-mail: jewell@us.ibm.com

The developers of PEMM had come to believe that the best way to address system performance was to proactively manage it throughout the solution development life cycle, from the system's inception through its deployment and maintenance in a production setting. It has been IBM's experience that throughout the last twenty five years; the principles within the PEMM have been long lasting and continue to deliver high value.

In an information technology (IT) context, *performance* generally refers to the speed with which a system[1] accomplishes the tasks it was designed to do, and is commonly expressed using measurements such as response time and throughput. Performance is closely related to and often dependent upon *capacity* (the measured ability of a system to perform work) and *scalability* (the ability of a system to accommodate workload growth). The dilemma faced by the IT industry with respect to performance and capacity is at least two-fold:

1. While systems, the technology on which they are based and the demands placed upon them are leading to increased complexity and therefore greater risk of poor performance, software engineering as a whole has traditionally been focused on ensuring the *functional correctness* of those systems rather than addressing performance, capacity and scalability concerns.
2. Even when so-called best practices for performance are followed during a system's development, it is difficult to grasp the full extent of inherent performance issues until all the components of the system can be integrated and tested to see how they perform together.

This has led to the situation where the industry has been working backwards over time to get a better handle on system performance. Poor performance in production led to the recognition of the value performance testing prior to deployment to identify and address performance issues. This in turn has led to focus on performance best practices, the notion of designing for performance, estimating and modeling in attempt to predict future performance, and creating requirements which unambiguously describe the users performance requirements and clearly link them to the needs of the business. The intent of the PEMM is to organize these activities in a cohesive manner, and to successfully manage performance requirements and mitigate risks throughout the software development life cycle.

[1] Throughout this paper, *system* refers to an interrelated group of IT components working together; *solution* refers to the application of IT systems, products, etc., to solve a customers business problem and/or address a customers business opportunity.

2.2 What is Performance Engineering?

Performance Engineering (PE) is a technical discipline which aims to ensure that a development project results in the delivery of a system which meets a prespecified set of performance objectives [2]. This is done by:

- managing the performance risk of a project,
- controlling or coordinating activities in the project that have an impact on performance, and
- applying specialized performance estimation and design skills to the architecture of the system under development.

In order to properly apply PEMM, it is important to understand the reasoning behind this view of performance engineering.

- **Performance objectives should be specified prior to development of the system.** A system that must consistently furnish subsecond response time or support 10,000 concurrent online users may be more expensive and time-consuming to design, build and support than an identical system with less stringent performance objectives. Requirements related to scalability, performance, capacity and scalability should be given the same level of attention given to the system's functional requirements when considering architecture, design and implementation alternatives.
- **Performance risks must be managed rather than eliminated.** It is generally not feasible to eliminate *all* risk of poor performance of a future system. The best we can hope to do is to find a cost-effective balance between performance *risk containment* (i.e. PE) activities and performance *risk acceptance*.
- **Achieving good performance requires coordinated effort.** Since so many groups and individuals are involved in defining, designing, implementing and deploying systems, typically there will not be just one group that makes or breaks performance. Coordinating PE activities throughout the life cycle will help to achieve the best possible results.
- **performance engineering is more than just waiting for performance testing to start.** There are a number of proactive PE techniques which can be applied even before performance testing starts in order to reduce performance risk. Proactive techniques such as performance modeling, performance budgeting, and application profiling can be used to get an idea of what performance will be like before an integrated system is available for performance testing purposes.

[2] The term Software Performance Engineering (SPE) is sometimes used in some of the literature pertaining Performance Engineering (PE) [10, 11, 12], although PE as discussed in this paper refers to information technology solutions in general and not just software.

2.3 Overview of PEMM

The central notion of PEMM is that performance engineering activities must be linked to the relevant portions of the system development life cycle. Figure 2.1 shows the conceptual view of PEMM in the context of a "typical" IT solution development project.

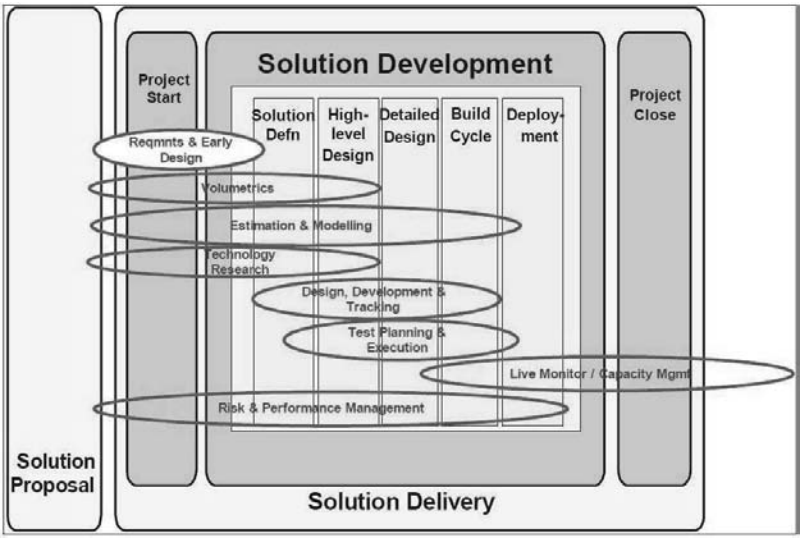


Fig. 2.1 Conceptual View of the Performance Engineering and Management Method (PEMM)

PEMM consists of eight interrelated, overarching "themes" which are active at various stages of the development project. The vertical bars depict stages which could be expected in a typical development project (proposal, project start, definition/requirements, design, build/test, deployment, project close). While the solution development process itself can take place with or without the use of PEMM, the ovals in the diagram above illustrate the relative chronology of the PEMM themes with respect to the solution development stages.

The PEMM themes can be described as follows:

- **Requirements and Early Design** - Performance activities occurring at the start of the development life cycle;
- **Volumetrics** - Quantitative elements of the workload and its data are vital to establishing the basis for the performance of the system;
- **Technology Research** - Gaining understanding of new solution elements (hardware, software, network, etc.) for which limited performance information is available;
- **Estimation and Modeling** - The central core of the predictive work done in Performance Engineering;

- **Design, Development and Tracking** - Helping the designers and developers deliver on the performance objectives;
- **Test Planning and Execution** - Testing to validate that the system will meet its performance and capacity objectives;
- **Risk Management / Performance Management** - Assessing and managing of the performance risks throughout;
- **Live Monitoring and Capacity Management** - Managing system performance and capacity for the rest of its operational life.

Each of these themes will be discussed in more detail in the chapters to follow. By applying the PEMM themes in a holistic manner throughout the development life cycle, the potential exists to

- improve system performance,
- reduce or more effectively manage system capacity and related costs,
- reduce or more effectively manage project and business risks related to performance,
- reduce the cost of addressing performance defects or issues by finding them and fixing them earlier.

2.4 PEMM Theme – Requirements and Early Design

Figure 2.2 illustrates some of the key considerations associated with the "Requirements and Early Design" theme.

Requirements and early design activities may require a certain amount of negotiation and iteration to come to completion. IT architects work with business analysts to document solution requirements. These requirements not only describe the system's desired capabilities, but also the dependencies and constraints (technical, financial, etc.) being imposed on the system. The architecture team will then consider one or more candidate solutions and evaluate whether they can meet requirements. Requirements that cannot be met may be renegotiated before going forward. This process is important not only for establishing a strong *technical baseline* [3] for change control purposes, but also for setting realistic expectations on the part of stakeholders.

While all of this is true for the general case of requirements and early design, there are additional considerations that are worth mentioning in regard to performance.

[3] Baseline documents include requirements documents, design documents and other materials defining the current state of the solution under development that have been reviewed and accepted by the projects business and technical stakeholders. Changes to baseline documents are carefully managed to prevent uncontrolled impacts to the project costs, quality and schedule.

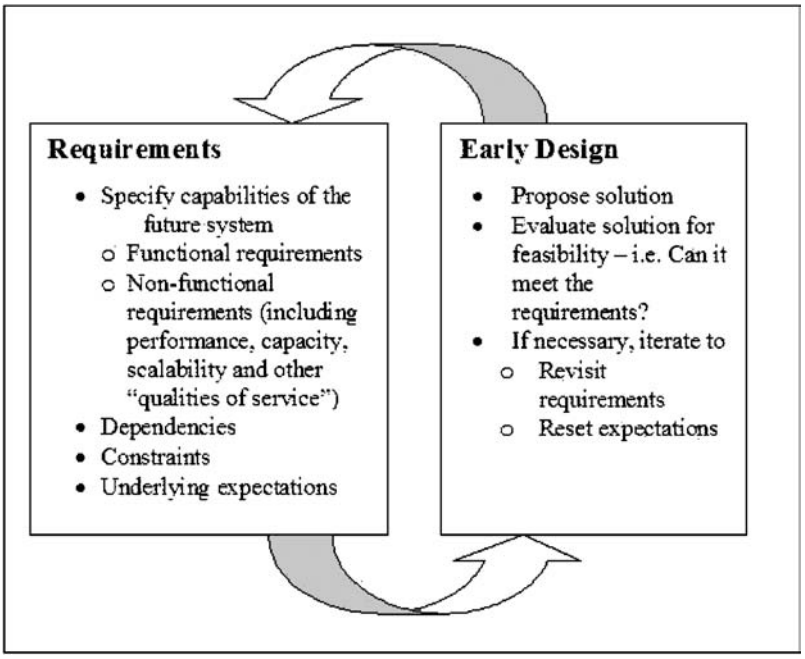


Fig. 2.2 Requirements and Early Design theme considerations.

2.4.1 Requirements and Performance

In engineering circles both inside and outside of IT, it is widely accepted that functional requirements must be understood before design can proceed – in other words, "what" is to be built must be known before "how" to build is addressed. Whether starting with formal written requirements using a waterfall development methodology or elucidating requirements incrementally using incremental development techniques, it is well recognized that systems development must be driven by requirements.

The notion of requirements can also be extended to what are referred to as *nonfunctional requirements*, i.e. assertions about the operational quality of service (QoS) constraints and dependencies which extend beyond application functionality. Menasce et al. [5] lists some typical IT quality of service characteristics which may be included in nonfunctional requirements:

- Response time
- Throughput
- Availability
- Reliability
- Security
- Scalability
- Extensibility

Architects and designers of IT systems attempt to choose the technical solution alternative which best satisfies all of the requirements, which in some cases may mean making trade-offs between requirements that may lead in different directions. For example, certain high availability measures may be more resource-intensive and impact response time or throughput. Behind some solution requirements may be contractual commitments, unstated expectations and other factors. An important first step towards achieving good performance is to ensure that performance requirements are sufficiently well documented and defined for each stage of the development process, and that they are clearly tied to the actual needs of the business. Vague or unrealistic performance requirements will hamper efforts to make the right technical decisions when trade-offs relating to performance must be made.

An important notion here is that in development projects, even the requirements themselves have their own engineering process associated with them. Requirements are to be written so that they are "SMART" (i.e. Specific, Measurable, Achievable, Realistic, Testable), as the architecture, design and build process will eventually decompose the high-level requirements into more granular components (system requirements, performance objectives, performance budgets – see also [9]). Throughout the development process, *requirements traceability* should be maintained so that all subsequent activities are clearly linked to the original requirements. While this is often done for functional requirements, this is sometimes neglected for performance and other nonfunctional requirement areas. PEMM helps ensure that nonfunctional requirements are addressed.

2.4.2 Early Design and Performance

The "Design for Performance" philosophy asserts that system performance cannot necessarily be taken for granted; therefore, when a certain level of performance is required, part of the design effort should be concerned with how the performance requirement will be met. While exact predictions of performance may be impossible to make during the architecture and high-level design stages of a project, to the extent possible, studies of the feasibility of the solution with respect to performance and other factors should be conducted.

Performance or capacity estimates made as part of an early feasibility study may be made with spreadsheets, formal sizing tools or comparisons to previously built systems. While they may not have the accuracy that future estimates would have when more is known about the solution, feasibility estimates provide a means of managing performance risk by evaluating design alternatives in terms of their effect on performance. If an area of performance risk can be flagged early, the requirements and business case can be revisited, and other technical solutions can be considered before going forward. While not all performance risk can be removed this way, feasibility estimates can disclose areas of performance risks that can be revisited in later stages of the development process.

2.5 PEMM Theme – Volumetrics

Figure 2.3 illustrates some of the key considerations associated with the "Volumetrics" theme [4]. As stated before, volumetrics refers to the quantitative elements of the workload and its data that are relevant to the performance and capacity of the system being built. Volumetrics can be considered as being in two categories: Business volumes and Technical volumes.

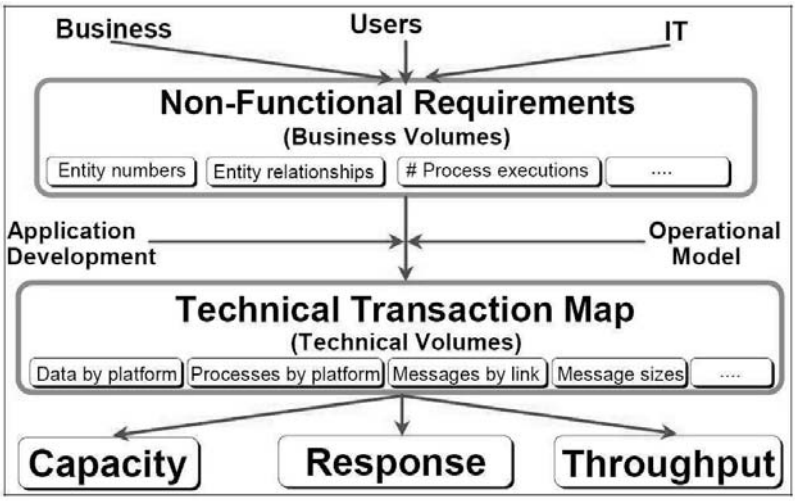


Fig. 2.3 Volumetrics theme considerations.

2.5.1 Business Volumes

Business volumes provide a business view of the quantitative elements that are important for planning system performance and capacity. These may include projections or assumptions for things such as the number of orders per day or week, the number of customers in the database and so on. At this level, the volumes are stated in terms that are relevant to the business, and may require someone with access to authoritative projections for business growth, etc., to be rendered accurately. Busi-

[4] Figure 3 includes a reference to an IBM architectural deliverable known as an Operational Model. The Operational Model (OM) defines and documents the distribution of an IT system's components onto geographically distributed nodes, together with the connections necessary to support the required component interactions, in order to achieve the IT system's functional and nonfunctional requirements within the constraints of technology, skills and budget.

ness, user and IT personnel may all be involved in developing the business volumes portion of the nonfunctional requirements.

Business volumes are often included with the nonfunctional requirements because they provide a critical context to the architects and designers of the system. For example, two systems with identical functionality may both have a mean response time requirement of 5 seconds, but if one of the systems only needs to support 300 concurrent online users while the other must support 30,000 concurrent online users, the design and capacity attributes of those two systems may need to be substantially different from each other. Similarly, the number of a key entity such as customer affects the amount of data to be processed or searched, ultimately affecting the responsiveness and resource intensiveness of data processing operations.

When developing business volume assumptions, the requirements team should take into account factors such as:

- whether a given volume element is likely to be static (not changing) or dynamic (changing over time)
- patterns of change in a dynamic volume element
- whether peak, average or point-in-time volumes are the most meaningful way to express a given volumetric's performance requirements
- the relationship between volumes and requirements for scalability (i.e. the ability to accommodate additional workload or data over time, given assumptions relative to resource capacity growth)

Business volumes are notoriously unstable, due to a number of factors. Business projections which support the volumetrics may be at best well-informed "guesses" as to the behavior of an unpredictable business climate. Projections may be overly optimistic or inflated to justify projects that would not otherwise have a viable business case. On the other hand, they may be overly low in an effort to contain costs. For critical volumes, it may be wise to come up with a low-to-high range of estimates so that options for "scaling up" or "scaling down" the solution can be considered. Corresponding assumptions should be well documented.

2.5.2 Technical Volumes

Technical volumes provide an IT solution view of the quantitative elements that are important for planning system performance and capacity. While technical volumes are based on the business volumes, they also take into account knowledge about the structure and behavior of the proposed solution gleaned from working with system architects, designers and developers.

Technical volumes are often captured in spreadsheet form using what is sometimes referred to as a technical transaction map. As a simplified example, assume that the business volumes reveal that there will be 10,000 orders per peak hour submitted on the order processing system. If the system interaction diagram reveals that each order produces:

- 1 order submit message from the client to the web server,
- 1 order submit message from the web server to the application server,
- 3 database request messages from the application server to the database server,
- 25 message database response messages from the database server to the application server,
- 1 order confirmation message from the application server to the web server,
- 1 order confirmation message from the web server to the client,

it can then be concluded that in a peak hour, order submit activity alone will generate:

- 20,000 incoming messages to the web server,
- 260,000 incoming messages to the application server,
- 250,000 incoming database requests to the database server,
- 10,000 outgoing message from the web server to clients.

In this example, it is clear that from the standpoint of the number of messages which must be handled for order submission traffic, the greatest message processing workload will fall to the application and database servers. Depending on the needs of the project, the technical transaction map can be expanded to include the volumes of data transferred and processed, message sizes and any other information that may prove relevant to estimating or modeling future performance and capacity.

2.6 PEMM Theme – Estimation and Modeling

Figure 2.4 illustrates a typical approach to the "Estimation and Modeling" theme. Estimation and modeling techniques are used to attempt to predict the performance and capacity behaviors of potential future systems, or of existing systems using potential future scenarios. The ideal means of estimating system performance is obviously to observe and measure the performance of a live system. When this cannot be done it may be necessary to develop a performance model (an abstract representation of the performance behavior of the future system) for this purpose.

Volumetrics, which have already been discussed, obviously constitute a key input to the performance estimation process. In addition, performance estimation must take into account other factors, such as:

- **Parametric costs** - The amount of a given system resource (CPU seconds, I/O operations, etc.) consumed per transaction, message or other estimating unit of work processed
- **Resource model** ? The computing hardware and other infrastructure elements which affect overall system performance, with capacity and processing power assumptions stated
- **Queuing model** - The means of predicting wait time (i.e. how long transactions are likely to wait for busy system resources to free up)

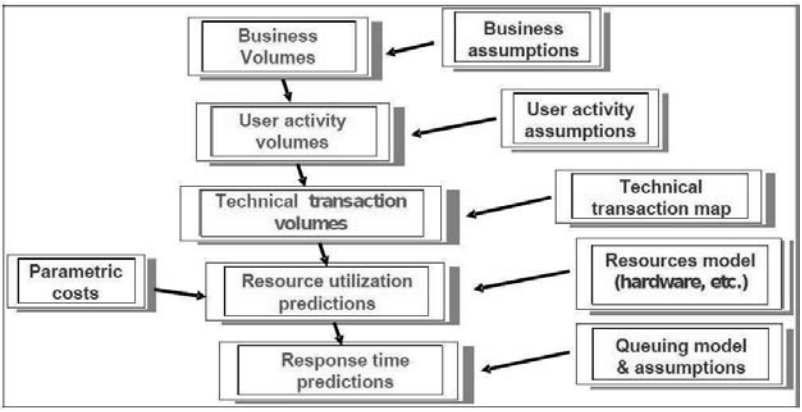


Fig. 2.4 Typical Estimation theme approach.

2.6.1 Performance Estimating Techniques

One of the more important aspects of performance engineering is the selection of the appropriate estimating technique for the situation at hand. Figure 2.5 shows some of the common estimating techniques and the corresponding cost vs. accuracy trade-offs. From a technical perspective, the difference in the techniques comes from the manner in which volumetrics, parametric costs, the resource model and the queuing model are represented and used to estimate performance.

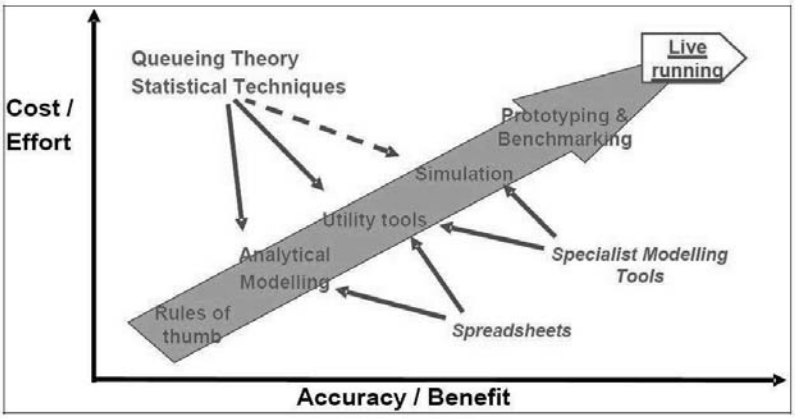


Fig. 2.5 Cost/Effort and Accuracy/Benefit trade-offs between performance estimation techniques .

Typically, more than one of these estimation techniques should be used during the life of the project. Low-cost, low-effort methods are commonly used early in the

project for feasibility study purposes, whereas the higher-accuracy methods may be employed later, when more is known about the solution design and validation of the solutions performance characteristics is deemed critical.

These estimating techniques can be summarized as follows.

- **"Rules of thumb" estimating** relies on very preliminary, simplified assumptions concerning volumetrics, parametric costs, system resources and wait times in order to deliver an estimate relatively quickly.
- **Analytical modeling** uses spreadsheets (or special purpose tools [5] in some cases) performing static calculations to make predictions of "steady state" performance and utilization of a system under a given workload. The static calculations typically take into account some amount of queuing theory and statistical techniques, in addition to volumetrics, parametric costs and system resources.
- **Utility tools** utilize data gathered from multiple performance benchmark results to facilitate the capacity sizing of systems based on reference workload types [6]. While such tools do not enable the creation of custom performance models for specific future systems, they do allow for limited specifications of volumetrics and system resources, and the parametric costs are gleaned from benchmark data tied to the reference workloads, allowing the sizing to be based on a similar workload to that of the target solution.
- **Simulation modeling** typically utilizes what is known as a discrete event simulation tool [7] to mimic the transaction processing behavior of the live system from a resource utilization and timing perspective. With these kinds of tools, the modeler must spend a considerable amount of time populating the model with the resource model, the parametric costs and transaction processing behavior (e.g. flows between the application tiers). Once this is done, the modeling tool simulates both the incoming transaction activity and the queuing/flow activity of the system being modeled, including the interaction between system components. This allows the dynamic performance behavior of the system over time to be examined.
- **Prototypes** can be thought of as partially built or preliminary versions of a contemplated invention, product or solution. The purpose of building any prototype is to learn from and leverage the experience of building or testing the prototype before making a commitment to the production version of whatever is being produced. For IT solutions, performance prototyping can serve as a means of investigating aspects of solution performance (e.g. estimating parametric costs) before a fully developed live system is available for performance testing.
- **Benchmark testing** as described here refers to the process of performance testing using a known workload before and after making a change or enhancement

[5] Accretive Technologies [1] is one vendor that offers analytical modeling tools.

[6] Examples of utility tools from IBM include the OPERA and SONOMA tools [2].

[7] A number of vendors (including HyPerformix and OpNet) and academic sources (e.g. Ptolemy II from Berkeley) offer discrete event modeling tools [3, 7, 8]. IBM Research has also developed an experimental performance modeling tool called AMBIENCE, which includes both analytical and simulation modeling capabilities. AMBIENCE also uses an inferencing engine to populate the initial model with starting parametric costs based on performance test results [4].

to the system, in order to determine whether the system's performance has been impacted by the change.

2.6.2 Selection of Performance Estimating Methods

Figure 2.5 has already introduced the general notion that increased accuracy in performance estimation comes only with increased cost and effort. We should not automatically reject the most accurate methods because of their expense or the least expensive methods because of their potential for inaccuracy. Performance estimation techniques, as is the case with most other performance engineering techniques, are intended to help us manage risks related to performance and capacity, so we must find ways to effectively balance the cost of accepting risk with the cost of containing those risks. The following recommendations are offered to help in making this selection.

1. **Consider how quickly an estimate is needed.** If an estimate is needed sooner rather than later, rough estimates and other low-effort techniques may be acceptable. This is particularly true in the early stages of a project, where the initial concern is to evaluate overall solution feasibility.
2. **Consider when the information needed for an estimate will become available.** If a certain estimating technique requires information that is not yet available, that technique may need to wait until later.
3. **Estimating techniques are not necessarily mutually exclusive.** While it may be tempting to use performance modeling in lieu of performance testing to reduce expense, the abstract nature of any performance model means that its accuracy has limitations. Rather, use performance modeling to help highlight the areas of risk that performance testing should focus on, and harvest the performance testing results as a means of calibrating the performance model for future use and achieving greater accuracy.
4. **Consider the return on investment.** Simulation modeling and performance testing often require significant investment in tools, training and development to be beneficial. Additionally, performance testing requires investment in a "production-like" hosting environment and test hardware. For these techniques in particular, there generally needs to be a return on investment in the form of reuse of the modeling and/or testing assets over time in order to justify the investments. Estimation efforts that will result in one-time, "throw-away" work may be better off using less expensive estimation methods.
5. **Consider the risk and criticality of performance for a given project.** Where poor performance has serious implication or the risk of poor performance is high, additional investment in performance estimating may be justified. Consider using performance estimation approaches which highlight likely risk areas, targeting those for additional estimation, testing and measurement.

2.7 PEMM Theme – Technology Research

A great deal of performance engineering requires dealing with uncertainty. This is certainly true of the "Technology Research" theme. Technology research is used here to describe techniques for gaining understanding of new solution elements (hardware, software, network, etc.) for which limited performance information is available. Frequently, the process of finding the data needed to populate a performance model (e.g. with volumetrics, parametric costs, etc.) ends up driving much of the technology research which takes place.

Typical approaches to technology research include the following.

- **Compare the future system with similar systems that already exist.** This is potentially one of the best sources of data available, particularly if the existing system is comparable in terms of the execution platform, volumetrics and so on.
- **Use published benchmarks to get an idea of the relative performance of solution alternatives.** Publicly available benchmarks are published by independent standards organizations such as TPC or SPEC, or in some cases by vendors, as a means of assessing solution alternatives using a common standard (e.g. to compare the relative processing power of two different server models). A benchmark usually involves a reference workload to be executed and a set of measurements to be gathered. Benchmarks may be executed either by vendors or independent test organizations that have an interest in evaluating or publicizing benchmark results for competitive evaluation purposes. The key thing to keep in mind is that benchmark results are only useful to the extent that the benchmark workload is representative of the projected system workload being modeled.
- **Consider using information from other reliable sources.** These may include published books, consulting reports, manufacturer specifications, Internet sources, or people with relevant experience.
- **Consider technical prototypes when needed information is difficult to get by other means.** As discussed earlier, performance prototyping involves building a partial solution to estimate performance attributes of the future solution.
- **Gather measurements from the system under development.** While this information will not be available at project planning time, information gleaned by development may prove very useful for refining existing performance models.

2.8 PEMM Theme – Design, Development and Tracking

Even if performance requirements have been well constructed, the solution is well architected and performance estimates show that the performance goals are attainable, ensuring that all of this comes together at implementation time is the responsibility of the development team. Performance considerations should be taken into account during detailed design and coding, and progress against performance goals should be validated as components become available for development testing.

From a process perspective, development activities are often carried out by someone other than the performance engineer or architect who helped to establish the original solution requirements, solution architecture and performance goals. Ensuring that continued progress is made towards meeting these goals requires close cooperation between all concerned. Fortunately, there are a number of approaches which can be successfully applied to address performance at implementation time.

2.8.1 Recognizing Performance Patterns and Anti-Patterns

Connie Smith and Lloyd Williams [9] described the notion of *performance principles, performance patterns* and *performance anti-patterns*. Table 2.8.1 illustrates each of these concepts in greater detail.

Concept	Description	Example
Principle for good performance	A general rule for creating designs that help achieve performance objectives	The principle of locality states that actions, functions and results should be located close to the computer resources, in order to reduce processing overhead.
Performance pattern	A frequently used design approach that tends to yield good performance results because it successfully exploits one or more principles of good performance	The alternate paths pattern helps improve the performance of data networks by providing multiple ways to send TCP/IP traffic from point A to point B, thus relieving network congestion.
Performance anti-pattern	A frequently used design approach that tends to yield poor performance results because it is inappropriately applied, or because it violates one or more principles of good performance	The one-lane bridge might be an acceptable solution for crossing a creek on a secluded country road, but would be an anti-pattern for a major highway crossing a river, or by analogy, for a low-bandwidth connection used where a high-bandwidth connection is needed.

Table 2.1 Performance Principles, Performance Patterns and Performance Anti-Patterns explained.

Smith and Williams list several principles, patterns and anti-patterns relating to performance, and quite likely more instances of each could be listed. The point is that a key to designing IT solutions that perform well is the understanding of design patterns that either do or do not work well from a performance perspective, and why that may be the case. Ultimately, performance is only one of several considera-

tions that IT architects and designers must take into account as they create solution designs. However, an understanding of performance patterns and anti-patterns will help in making the inevitable trade-offs between performance goals and other solution requirements.

2.8.2 Designing for Performance

Software engineering practitioners have long recognized the need to design for functionality, and accordingly, software quality techniques and best practices have been instantiated again and again, taking the forms of waterfall development, structured programming, iterative development, object-oriented development, agile development methods and so on. The means of designing for performance, however, have arguably been less conspicuous as part of software engineering practice.

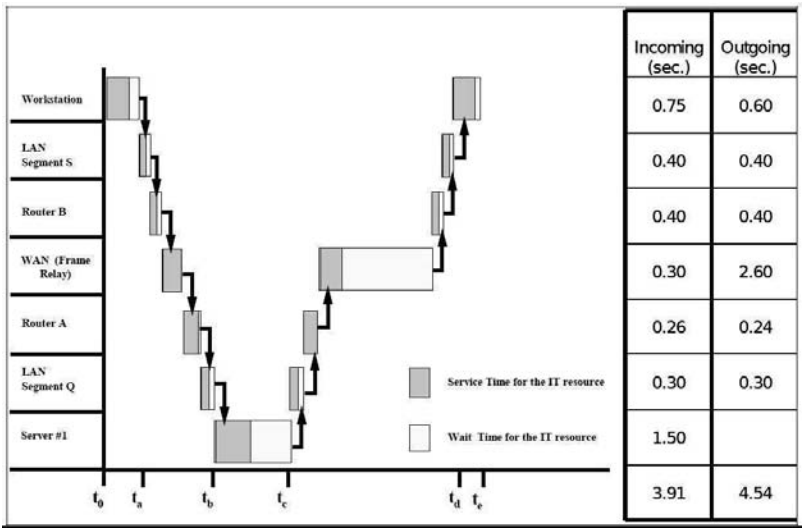
The "Design for Performance" philosophy is one that takes a goal-oriented approach to developing IT solutions that perform well, rather than taking it for granted that everything will somehow get worked out in the end. This involves more than just having programmers keep a handy checklist of Java and SQL performance "dos and don'ts". "Design for Performance" essentially means that performance goals or targets are consciously set, and design and implementation decisions are made and validated with respect to their effects on performance, in such a way that performance does not become a "nasty surprise" at the end of the project.

As much as possible, this means that design decisions creating good performance reduce the need to make "tuning" changes to the implemented solution after the fact. Applying the correct performance patterns, avoiding the correct performance anti-patterns, leveraging the advanced performance capabilities of the implementation platforms and then validating performance throughout the development process will go a long way towards making our performance engineering activities more proactive and less reactive.

2.8.3 Performance Budgeting

As with functional requirements, initial performance requirements are typically defined at a high level for business and end-user audiences. However, in order to be useful for tracking at the development level, they must eventually be decomposed into more granular components as the solution design takes shape. These more granular, component-level goals for performance are often assembled into what are known as performance budgets. A performance budget allocates time or system resources (not money!) required to complete a task for planning and estimation purposes. Figure 6 shows an example of a response time performance budget.

Initially, a response time budget such as this one might be based on estimates of the time spent processing a transaction at each layer of the infrastructure. Eventually,



Incoming (sec.)	Outgoing (sec.)
0.75	0.60
0.40	0.40
0.40	0.40
0.30	2.60
0.26	0.24
0.30	0.30
1.50	
3.91	4.54

Fig. 2.6 Example of a simple response time budget.

through the use of some kind of tracing or instrumentation, actual times for each of these components can be gathered and compared to the original budget. At that time, the process for staying in budget is similar to one that would be used on a monetary budget; developers would look for opportunities to reduce or eliminate unnecessary processing and/or make modifications to the budget that reflect the current reality. If this meant that a critical response time target was not being met after applying potential tuning changes, either the solution design or the performance requirement would need to be revisited.

2.8.4 Performance Debugging and Profiling

Developers often have available to them debugging and profiling tools that can help isolate performance problems. While the term debugging has long been associated with isolating and removing functional defects from application code, many debugging tools and techniques can help improve performance as well. For example, thread contention and memory leaks are examples of functional defects which can be found using debugging tools and techniques which also serve to improve performance.

Profiling tools and techniques, on the other hand, are generally involved with measuring application and/or subsystem performance. In particular, profiling tools and techniques are especially helpful for

- locating and resolving performance inefficiencies,
- isolating problems in a vertically large multi-tiered architecture,

- optimizing performance at the code, database or network level,
- determining whether performance budgets can be met.

2.8.5 Design, Development and Tracking Guidance

As was the case with the "Estimation and Modeling" theme, the various approaches associated with the "Design, Development and Tracking" theme should be used judiciously. Just as it is not feasible to exhaustively test an application system of any significant size, it is not generally feasible to apply all of the techniques discussed here to the entire application being developed. For example, labor-intensive techniques such as performance budgeting and application profiling are typically reserved for areas of the application which are associated with use cases for which performance is critical and performance requirements are challenging. The balance between risk and containment may dictate how each of the "Design, Development and Tracking" approaches should be used.

2.9 PEMM Theme – Test Planning and Execution

Performance testing is concerned with ensuring that nonfunctional requirements having to do with performance, capacity and scalability can be met by the solution as implemented. The job of the performance tester is made much easier if there are clear performance requirements and a proactive "Design for Performance" philosophy has been followed in architecting, designing and implementing the solution. Starting with an understanding of the designed solution and its performance requirements, the performance tester must design, implement and execute performance testing scenarios which validate the ability of the solution to meet the performance requirements.

To understand performance testing, it is helpful to understand the distinctions between functional testing and performance testing as shown in Table 2.9.

As with other kinds of testing, performance testing requires the use of effective test management practices (e.g. planning, entrance and exit criteria, designing test cases that are traceable to requirements, test execution tracking, defect management). However, the focus on performance rather than function means that different tools, skills, methods and approaches must be brought to bear. Moreover, traditional test progress metrics such as percentage of test cases complete, defect counts and defect severity may not convey the seriousness of underlying performance problems which the performance testing is starting to reveal.

Table 2.9 illustrates a number of typical activities, deliverables and work products associated with performance testing.

Performance testing of this kind is not a novel industry concept, nor is it unique to PEMM. However, with PEMM we no longer depend exclusively on performance

Functional Testing	Performance Testing
Concerned with coverage and correctness of function	Concerned with responsiveness, throughput and behavior under heavy workload with limited resources
Validates behavior in response to executing individual test cases	Validates behavior in the operational environment in response to a representative "mix" of activity
Defects note incorrect behavior.	Performance deficiencies may be a matter of degree.
Test tools and skills are oriented towards the single-user perspective.	Test tools and skills are oriented towards multi-user, system perspective.

Table 2.2 Differences between Functional Testing and Performance Testing.

testing to manage risks related to performance. Applying the preceding PEMM themes in a proactive manner can go a long way towards averting performance issues before testing or deployment. Furthermore, by applying PEMM, the performance tester can

- benefit from having performance requirements to use in developing test scenarios,
- leverage the findings of earlier estimation and modeling work to help prioritize testing efforts, and
- share test results with the modeling team to help refine the predictive models for future use.

2.10 PEMM Theme – Live Monitoring and Capacity Planning

The "Live Monitoring and Capacity Management" theme is concerned with activities needing to take place to ensure that the solution performs well in the live production environment.

The "needs of the business" should dictate the business requirements for the system, including those nonfunctional requirements related to performance, capacity and scalability. While these requirements place certain demands on the teams responsible for building the solution, they also place demands on the teams responsible for deploying, hosting and operating the solution in its production environment. For example, capacity plans must be developed and validated to ensure that the infrastructure is ready to handle the workload. Tools and procedures (preferably auto-

Test Process Stage	Activity	Deliverables and work products
Test Planning	Analyze requirements	• Nonfunctional requirements • Volumetrics
	Determine test strategy	• Selected use cases / business functions • Infrastructure scope • Application components • Monitoring tools • Test process, including phases and scenarios
	Create test plan	• Test plan
Test Preparation	Develop workload	• Workload design • Workload scripts • Workload procedures, execution plan
	Develop measurement procedures	• Measurement procedures
	Set up environment	• Prepared test environment, agreement for operations support
Test Execution	Run tests	• Test run output, measurement data • Test progress reporting
	Analyze results	• Interim results reports
	Follow-up	• Defect documentation • Problem resolution correspondence
Test Reporting	Prepare results report	• Results reports by test phase
	Publish and review results	• Review minutes
	Decision checkpoint with stakeholder	• Sign-off by phase • Deployment recommendation and decision

Table 2.3 Performance testing stages, activities, deliverables and work products.

mated) must be put in place to respond to poorly performing applications and transactions. Furthermore, there may be service level agreements between the provider (operations) and consumer (users) of IT services, in order to set expectations concerning the projected workload and resulting performance of the live production system.

2.10.1 Relating PEMM to Performance and Capacity Management

Performance management has to do with ensuring that the deployed production system performs as required, and taking corrective action when it does not. Closely related to this is *capacity management*, which has to do with planning for, deploying and monitoring physical computing resources and infrastructure to ensure that the system has sufficient resources to do its job.

Performance Management and Capacity Management of live, operational production systems are not new concepts. For years, data center operations professionals have understood that these and other systems management disciplines (problem management, change management, etc.) as being essential to running things smoothly in the production environment. In more recent years, these disciplines have become part of what is now known in the industry as *IT Service Management*, with frameworks such as the *IT Infrastructure Library (ITIL®)* beginning to codify the industry consensus on the leading practices in this area [6].

Figure 2.7 illustrates the key IT service management processes encompassed by ITIL Version 2 [8]. Note that in ITIL terminology, both performance management and capacity management are addressed by the ITIL Capacity Management process.

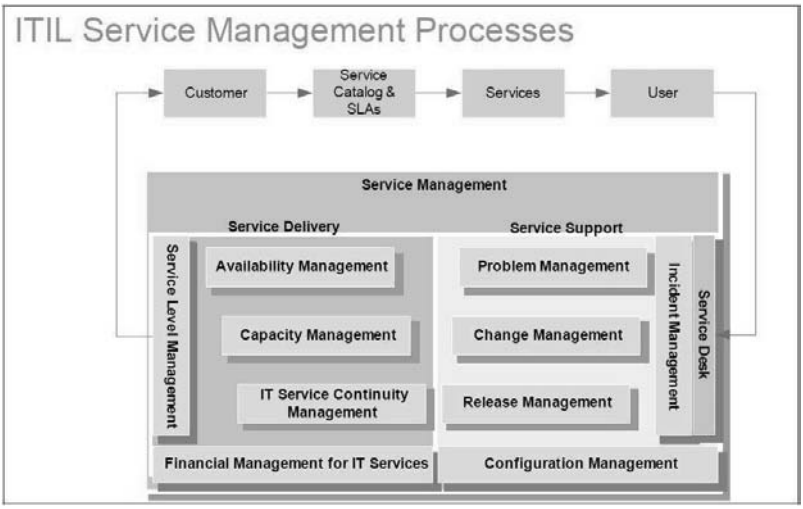


Fig. 2.7 Overview of ITIL Service Management processes.

While PEMM is not overly prescriptive about how performance and capacity management ought to be done, the chief PEMM recommendation in this area is that the potential for synergy between Performance Engineering efforts and performance / capacity management (PCM) efforts should be leveraged. This is not

[8] As of this writing, the latest version of ITIL is Version 3. However, the comments concerning the key ITIL service management processes still apply.

always as easy is it would seem to be. In many IT organizations, the differing skills, responsibilities and locations of the development and operations shops prevent the respective groups from working together, and what interfaces there are take the form of required meetings and paperwork hurdles needed to get final approvals for deployment. However, looking at the PEMM themes from a life cycle perspective, it is not hard to see potentials for achieving synergy with and improving hand-offs to the PCM team. For example:

- The "Requirements and Early Design" theme can contribute to the definition of candidate *service level requirements* (SLRs), which may become the basis for later service level agreements and for some of the development performance targets.
- The "Estimation and Modeling" theme can and should contribute to capacity planning efforts.
- The "Test Planning and Execution" theme can include exercising the performance monitoring tools and procedures that will be used in production, and providing final guidance with respect to performance and capacity expectations prior to deployment.

Achieving these kinds of synergies may become a requirement for organizations seeking to fulfill the spirit and intent of industry initiatives such as ITIL Capacity Management and the full solution life cycle scope of PEMM.

2.10.2 PEMM and ITIL Capacity Management

The ITIL service management framework has provided the industry with a widely accepted framework and vocabulary for understanding and implementing service management practices. Capacity Management is one of the major service delivery processes described in ITIL, and is defined rather broadly to address not only the capacity of IT systems and infrastructure, both also the capacity of other resources (facilities, human resources, electrical power) on which the delivery of IT services may depend. ITIL Capacity Management defines three major subprocesses:

- **Business Capacity Management**
  - Manage Service Level Agreements (SLAs) and future Service level Requirements (SLRs)
  - Study Business Plans & Create resultant Capacity Plans
  - Perform Modeling & Application Sizing for Business Services
- **Service Capacity Management**
  - Translate Business SLAs & SLRs into IT SLAs & SLRs
  - Measure Throughput & Performance for System, Network & Service
  - Perform Monitoring, Measuring, Analysis, Tuning & Demand Management

- **Resource Capacity Management**
  - Optimize & Configure Current Technology
  - Research Future Technology and other alternatives
  - Guarantee System and Service Resilience

There is a sequential flow inherent to this arrangement of the subprocesses.

1. **Business capacity** – How much business work must the solution perform?
2. **Service capacity** – How much work must the IT services do to meet the business capacity requirements?
3. **Resource capacity** – What physical resources, etc., are needed to address service capacity requirements?

Moreover, most (if not all) of the activities supporting the ITIL Capacity Management subprocesses are essentially performance engineering activities as described in PEMM. Because PEMM and ITIL Capacity Management originate from different perspectives (development vs. operational), they are actually complementary in many respects, and can be used effectively together.

2.11 PEMM Theme – Performance and Risk Management

Figure 2.8 illustrates some of the key considerations associated with the "Performance and Risk Management" theme. This theme is concerned with activities needing to effectively manage performance-related risk throughout the life of the project.

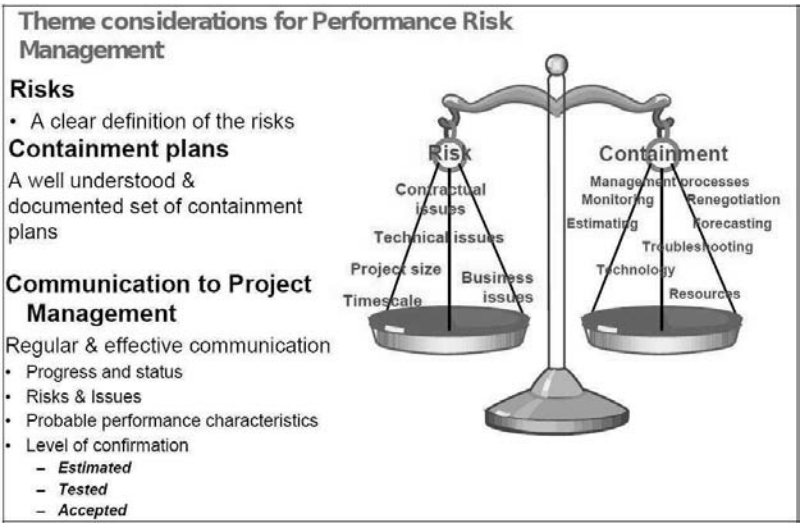


Fig. 2.8 Key considerations for the "Performance and Risk Management" theme.

Managing performance-related risks is one of the chief overall objectives of PEMM. Nearly every one of the other more "technical" PEMM themes is intended to help reduce risks related to the potential for poor performance. Just as a project manager is responsible for the management of risks to the overall project, the performance engineer must be concerned with identifying and mitigating technical risks related to performance, capacity and scalability, by engaging the right resources and/or management attention when significant performance issues or risks arise. This is done with the recognition that since it is not feasible to eliminate all sources of performance-related risks, *finding an effective balance between risk containment and risk acceptance* should be the goal.

From a project perspective, there are other things that can also be done to facilitate the management of performance risk.

2.11.1 Assignment of Dedicated Performance Engineering Resources

For complex systems with critical performance requirements, it is important to establish accountability for the management and coordination of activities needed to achieve the project's IT performance goals. The terms *performance architect* or *performance engineer* are sometimes used to describe the role of those who assume this responsibility for a given project. Depending on the project, there may be a team of performance engineers sharing this responsibility.

While the size of the performance engineering team may vary depending on the complexity and the level of performance risk associated with the project, it is not unusual to see 1% to 5% of an IT projects total cost invested in performance engineering activities, or more for complex projects [9, 10, 11, 12]. While performance engineers are typically most involved in the earlier themes of PEMM, there is a great deal to be gained if there is active PE involvement throughout the solution life cycle. The Performance Engineer may not own direct responsibility for activities such as development, testing and post-deployment performance and capacity management; however, the PE must nevertheless find ways to maintain the influence and accountability needed to help the project meet its performance, capacity, and scalability requirements.

2.11.2 Applying PEMM to IT Project Governance

The need of the performance engineer to influence activities and decisions not under his or her direct control begs the question of how PEMM relates to overall IT project governance. Most IT organizations have some level of methodology and process documentation which describes how activities such as architecture, design, development testing, deployment and live operations are to be handled, along with

a governance structure which monitors and enforces adherence to these methodologies and processes. A prudent means of applying PEMM is not necessarily to replace existing processes, but rather to ensure time and resources are allotted to integrate PE activities into the existing plans, and to identify touch points between PEMM and the local process/methodologies to ensure that adequate progress is being made towards meeting the performance goals.

2.11.3 Applying PEMM to Complex Projects

Inevitably, projects will be encountered where it is not possibly to exhaustively apply PEMM techniques to all aspects of the solution. In such cases, one must be selective about the scope and focus of performance engineering efforts. Some considerations for managing this complexity from a PEMM perspective are listed below.

- **Divide and conquer** – Rather than trying to estimate or model all the aspect of a large system or group of systems in their entirety, it may make more sense to split them into more manageable subsystems for separate consideration.
- **Prioritize** – As was stated earlier, be selective about which parts of the system warrant detailed PE studies. Concentrate on those areas where the risk and/or potential impact of performance issues is most significant. Special consideration should be given in the following cases.
- **Special cases** – The following classes of solutions often have specific performance challenges requiring special attention.
  - **Multi-channel solutions** – If a central system is being accessed by multiple channels (direct web users, telephone VRU, call center users, etc.), be aware that each channel will have different effects on and requirements pertaining to system performance.
  - **Distributed solutions** – If a common solution is replicated at multiple data centers, keep in mind that there could be site-specific performance and capacity considerations depending on how the worldwide workload is distributed.
  - **Commercial Off-The-Shelf (COTS) Packages** – It is increasingly common for customers to use vendor-provided packaged solutions or frameworks to meet their specific business needs. Ensuring that a vendor solution continues to meet performance requirements after being configured, tailored or extended to meet the customers needs is often challenging.
  - **Large programs** – Some organizations may have major conversion or transformation initiatives that take months or years to implement. These situations may have governance and technical considerations that make PE challenging in those cases.
  - **Unpredictable workloads** – The advent of the web era has shown that web-enabled workloads originating from the Internet can be difficult to predict.

Making contingency plans for additional capacity and scalability is especially important in those situations.

- **Complex interactions** – Solutions that are based on SOA and other highly modular architectures can pose workload balancing issues and other performance challenges in their own right.

2.12 Summary

The Performance Engineering and Management Method is a collection of IBM's leading practices with respect to performance engineering and performance management of IT systems. PEMM is organized as a framework consisting of eight overarching and interrelated "themes" which operate in concert with whatever locally used IT solution life cycle is in place. While the themes themselves are not necessarily unique to IBM or to PEMM, the overall framework provided by PEMM helps to conceptualize the performance engineering activities that are appropriate for each stage of the solution life cycle.

Even when users of PEMM have not had the opportunity to apply all of PEMM to a given project, having an understanding of the PEMM themes still helps to understand what things should have been done previously and should be done now to manage performance-related risks during the project life cycle. Within IBM, acquainting architects, developers, testers, and operations personnel with the concepts of PEMM have raised awareness of both the potential for applying PE in a proactive manner and the potential for achieving synergies across the themes by working together.

Implementing PEMM is not without its challenges. Managers seeking to reduce costs and shorten schedules may need convincing before they are willing to dedicate resources to performance engineering activities, and even then the scope of PE activities may be limited due to the current state of the project. Performance estimating and modeling methods are arguably immature, and commercially available performance modeling and testing tools are sometimes prohibitively expensive. Even within IBM, the process of institutionalizing PEMM has been a gradual one, relying on educational offerings, methodology enhancements, project experience and an active internal PE advocacy community to encourage PEMM adoption over time.

Nevertheless, progress continues to be made. New training has been introduced to train IBM practitioners in applying PEMM to SOA projects. PEMM concepts have been applied by IBM in both internal and commercial projects for years, and now customer demand has led to the development of commercially offered PEMM training. Moreover, the continued acceptance of process maturity frameworks such as ITIL is leading to increased awareness among IT organizations that system performance must be treated as a business imperative, and not left to chance. Given this, PEMM and similar holistic approaches to PE will fulfill a vital need in the IT industry for years to come.

Acknowledgements The author thanks Martin Jowett, Damian Towler, Chris Winter, Ann Dowling, Zhen Liu and Cathy Xia for their review of and contributions to this paper.

References

1. Accretive Technologies home page. http://www.acrtek.com/acc_home.swf.
2. E. Hung, Q. He, and J. Zhu. Sonoma: Web service for estimating capacity and performance of Service-Oriented Architecture (SOA) workloads, 2006. ftp://ftp.software.ibm.com/software/dw/wes/hipods/SONOMA_wp9Oct_final.pdf.
3. HyPerformix home page. http://www.hyperformix.com.
4. Z. Liu, L. Wynter, C. Xia, and F. Zhang. Parameter inference of queueing models for it systems using end-to-end measurements. *Performance Evaluation*, 63:36–60, 2006.
5. D. A. Menasce and Others. *Performance by Design: Computer Capacity Planning by Example*. Prentice Hall, Upper Saddle River, NY, 2004.
6. Official ITIL® web site home page. http://www.itil-officialsite.com/home/home.asp.
7. OpNet home page. http://www.opnet.com.
8. Ptolemy II home page. http://ptolemy.berkeley.edu/ptolemyII.
9. S. C. U. and L. G. Williams. *Performance Solutions: A Practical Guide to Creating Responsive, Scalable Software*. Prentice Hall, Upper Saddle River, NY, 2002.
10. SPE Experience: An Economic Analysis. http://www.perfeng.com/papers/hesselg2.pdf.
11. The Economics of SPE Panel. http://www.perfeng.com/papers/jennings.pdf.
12. What Does Software Performance Engineering Cost? http://www.cs.ucl.ac.uk/staff/ucacwxe/lectures/3C05-01-02/aswe13.pdf.