Five Steps to Establish Software Performance Engineering in Your Organization

Connie U. Smith, Ph.D.
Performance Engineering Services
PO Box 2640
Santa Fe, New Mexico, 87504-2640
(505) 988-3811
http://www.perfeng.com/

Lloyd G. Williams, Ph.D.
PerfX
2345 Dogwood Circle
Louisville, Colorado 80027
(720) 890-8116
lloydw@perfx.net

Most software performance problems are due to fundamental architecture or design problems. Thus they are introduced early in development, but are typically not discovered until late, when they are more difficult and costly to fix. Software Performance Engineering (SPE) is a systematic, quantitative approach for cost-effectively building performance into software systems. It consists of a process for applying the SPE methods throughout the life cycle of new systems; data required for SPE studies; software and system execution models for quantitative performance assessments; and other techniques for success application to software projects.

This paper covers the essential steps for establishing an SPE capability in your organization. It describes each step and gives pointers to additional information that will help you implement them in your organization.

INTRODUCTION

Software performance engineering (SPE) is a method for constructing software systems to satisfy performance requirements. The SPE process begins early in the software life cycle and uses quantitative methods to identify satisfactory designs and to eliminate those that are likely to have unacceptable performance before developers invest significant time in implementation. SPE continues through the detailed design, coding and testing phases to predict and manage the performance of the evolving software as well as monitor and report actual performance versus specifications and predictions. SPE methods encompass: performance data collection; quantitative analysis techniques; prediction strategies; management of uncertainties; data presentation and tracking; model verification and validation; critical success factors; and performance design principles, patterns and antipatterns.

Experience shows that performance of new systems can be orders of magnitude better, with no disruption to delivery schedules, when these techniques are systematically applied throughout development. Performance is typically addressed, however, after performance problems are detected late in development. It is difficult to persuade managment and developers to apply SPE early enough in development to prevent problems.

Persuading organizations to apply SPE can be problematic. SPE is sensible and it works, so sometimes a technical argument may succeed. Other times, you may want to start with a business case to present the financial reasons and benefits (as covered in step 5). If you know of a recent “performance disaster” you may leverage the persuasion with a presentation of how to avoid similar situations in the future. The use of new technology increases risks of performance problems, so you may pitch the idea to manage those risks. Outsourcing the development of critical systems is no guarantee that performance requirements will be met unless you apply SPE during development, so that’s another opportunity to pitch the idea.

This paper starts at the point that you have justified the use of SPE and are ready to establish it. It covers five essential steps to establish SPE in your organization:

1. Learn the SPE process, modeling and analysis techniques
2. Create the initial team
3. Acquire an initial set of tools
4. Strengthen skills on a pilot project

5. Establish on-going SPE

In order to be self-contained, we first cover the steps in the SPE process and the models used to predict performance. Then we offer advice for establishing a team, tools, and organization, conducting a pilot project, and building a business case. The steps do not have to be applied in this order, you may adapt them to your particual situation. For example, some organizations require a business case before acquiring tools.

Step1: Learn the SPE Process and Modeling and Analysis Techniques

Before you start, you need a good understanding of the overall process, the quantitative models, the analysis techniques, and how to apply them. This will help you explain them to others to get initial buy-in, and give you a quick start on your first project. A brief overview is included in the following section. You can get more information by attending a class, attending conference presentations, and/or reading a book such as [Smith and Williams 2002].

SPE Overview

Performance is an essential quality attribute of every software system. Many object-oriented and non-object-oriented software systems, however, cannot be used as they are initially implemented due to performance problems. Systems delivered with poor performance result in damaged customer relations, lost productivity for users, lost revenue, cost overruns due to tuning or redesign, and missed market windows.

It is possible to cost-effectively design performance into new software systems. Doing this requires careful attention to performance goals throughout the life cycle. Software performance engineering (SPE) provides a systematic, quantitative approach to managing performance throughout the development process.

SPE uses deliberately simple models of software processing with the goal of using the simplest possible model that identifies problems with the system architecture, design, or implementation plans. It is relatively easy to construct and solve these models to determine whether the proposed software is likely to meet performance goals. As the software development process proceeds, we refine the models to more closely represent the performance of the emerging software and re-evaluate performance.

SPE is language and platform independent. The models are constructed from architectural and design-level information. Thus, SPE works with C++ and Java as well as with other object-oriented and non-object-oriented languages. The execution behavior of the software will be different with different languages and platforms. Nevertheless, this is reflected in the resource requirement specifications, not the model structure.

SPE can be easily integrated into the software development process. It has been used with traditional process models, such as the waterfall model. It works especially well with iterative, incremental processes such as the Unified Process [Kruchten 1999], [Jacobson, et al. 1999]. With an iterative, incremental process, you can use SPE techniques to assess and reduce the risk of performance failure at the project outset, and at each subsequent iteration.

The cost of SPE is usually a minor component of the overall project cost. Lucent Technologies has reported that the cost of SPE for performance-critical projects is about two to three percent of the total project budget. For other, less critical, projects, SPE typically costs less than one percent of the total project budget. For projects where the performance risk is very high, the SPE expenditure may be as much as ten percent of the project budget.

SPE efforts can save far more than they cost by detecting and preventing performance problems. Bank One reported that, on one project, SPE costs over a five-month period were $147,000. During this time, the team analyzed three applications and identified modifications that resulted in a projected annual savings of $1,300,000 [Manhardt 1998]. Similarly, the performance engineering group at MCI reported a $20,000,000 savings in one year with SPE due to reduced resource requirements that resulted in deferred configuration upgrades [CMG 1991]. Similar compelling results were reported in panel sessions at recent CMG conferences.

SPE is not a silver bullet or a cure-all for performance problems. SPE takes thought, effort, and analysis to produce the desired results. However, SPE efforts can save far more than they cost by detecting and preventing performance problems. The return on investment for SPE more than justifies its use.

SPE Process and Models

The SPE process is facilitated by object-oriented modeling notations such as the Unified Modeling Language (UML)[Rumbaugh, et al. 1999]. Much of the information needed for SPE's performance models can be captured as part of the object-oriented analysis and design process. Use cases, which are identified as part of the requirements definition, are a natural link between software development activities and SPE. The scenarios that describe the use cases provide a starting point for constructing the performance models.

The SPE process includes the following steps. The activity diagram in Figure 1 captures the overall process.

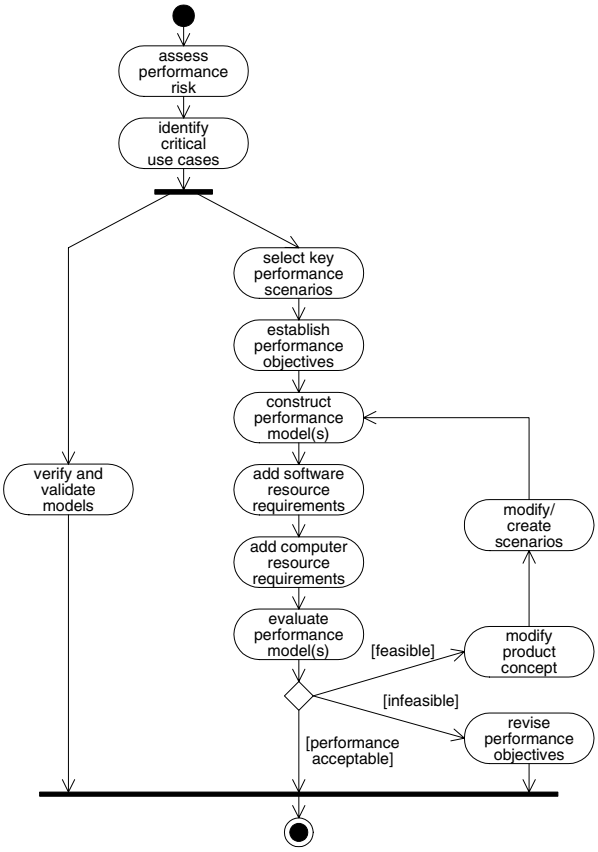

Figure 1: The SPE Process

1. *Assess performance risk*: Assessing the performance risk at the outset of the project tells you how much effort to put into SPE activities. If the project is similar to others that you have built before, is not critical to your mission or economic survival, and has minimal computer and network usage, then the SPE effort can be minimal. If not, then a more significant SPE effort is needed.
2. *Identify critical use cases*: The critical use cases are those that are important to the operation of the system, or that are important to responsiveness as seen by the user. The selection of critical use cases is also risk driven. You look for use cases where there is a risk that, if performance goals are not met, the system will fail or be less than successful. Typically, the critical use cases are only a subset of all the use cases of the software.
3. *Select key performance scenarios*: It is unlikely that all of the scenarios for each critical use case will be important from a performance perspective. For each critical use case, the key performance scenarios are those that are executed frequently, or those that are critical to the perceived performance of the system. Each performance scenario corresponds to a workload. Scenarios are represented by UML sequence diagrams.
4. *Establish performance requirements*: Next, identify and define performance requirements and workload intensities for each scenario selected in step 3. Performance requirements specify the quantitative criteria for evaluating the performance characteristics of the system under development. These requirements may be expressed in three primary ways: by response time, throughput, or constraints on resource usage. For information systems, response time is typically described from a user perspective, that is, the number of seconds required to respond to a user request. Throughput requirements are specified as the number of transactions or events to be processed per unit of time.

   Workload intensities specify the level of usage for the scenario. They are specified as an arrival rate (e.g., number of Web site hits per hour) or number of concurrent users.

Repeat steps 5 through 8 until there are no outstanding performance problems.

5. *Construct performance models*: We use execution graphs to represent software processing steps in the performance model. Step 3's sequence-diagram representations of the key performance scenarios are translated to execution graphs.
6. *Determine software resource requirements*: The processing steps in an execution graph are typically described in terms of the software resources that they use. Software resource requirements capture computational needs that are meaningful from a software perspective. For example, we might specify the number of messages sent or the number of database accesses required in a processing step.

   Early in the development process, estimates of resource requirements may be simple best- and worst-case estimates. Later, as each class is elaborated, the estimates become more precise.
7. *Add computer resource requirements*: Computer resource requirements map the software resource requirements from step 6 onto the amount of service they require from key devices in the execution environment. Computer resource requirements depend on the environment in which the software executes. Information about the environment is obtained from the UML deployment diagram and other documentation. An example of a computer

resource requirement is the number of CPU instructions and disk I/Os required for a database access.

Steps 6 and 7 could be combined, and the amount of service required from key devices estimated directly from the operation specifications for the steps in the scenario. However, this is more difficult than estimating software resources in software-oriented terms and then mapping them onto the execution environment. In addition, this separation makes it easier to explore different execution environments in "what if" studies.

8. *Evaluate the models*: Solving the execution graph characterizes the resource requirements of the proposed software alone. If this solution indicates that there are no problems, you can proceed to solve the system execution model. This characterizes the software's performance in the presence of factors that could cause contention for resources, such as other workloads or multiple users.

   If the model solution indicates that there are problems, there are two alternatives:
   - *Modify the product concept*: Modifying the product concept involves looking for feasible, cost-effective alternatives for satisfying this use case instance. If one is found, we modify the scenario(s) or create new ones and solve the model again to evaluate the effect of the changes on performance.
   - *Revise performance requirements*: If no feasible, cost-effective alternative exists, then we modify the performance goals to reflect this new reality.

     It may seem unfair to revise the performance requirements if you can't meet them (if you can't hit the target, redefine the target). It is not wrong if you do it at the outset of the project. Then all of the stakeholders in the system can decide if the new requirements are acceptable. On the other hand, if you get to the end of the project, find that you didn't meet your goals, and then revise the requirements—that's wrong.
9. *Verify and validate the models*: Model verification and validation are ongoing activities that proceed in parallel with the construction and evaluation of the models. Model verification is aimed at determining whether the model predictions are an accurate reflection of the software's performance. It answers the question, "Are we building the model right?" For example, are the resource requirements that we have estimated reasonable?

   Model validation is concerned with determining whether the model accurately reflects the execution characteristics of the software. It answers the question, "Are we building the right model?" We want to ensure that the model faithfully represents the evolving system. Any model will only contain what we think to include. Therefore, it is particularly important to detect any model omissions as soon as possible.

   Both verification and validation require measurement. In cases where performance is critical, it may be necessary to identify critical components, implement or prototype them early in the development process, and measure their performance characteristics. The model solutions help identify which components are critical.

These steps describe the SPE process for one phase of the development cycle, and the steps repeat throughout the development process. At each phase, you refine the performance models based on your increased knowledge of details in the design. You may also revise analysis objectives to reflect the concerns that exist for that phase.

More Information

This brief introduction covers only a small part of the SPE techniques. SPE is a broad discipline and this paper focuses on establishing a practice to apply the basic analytical aspects of SPE. There are many other facets of SPE, such as techniques for obtaining necessary data, analysis strategies, contention delays due to multiple users of a scenario and other workloads that may compete for computer system resources, advanced modeling techniques for Web and other distributed systems, performance principles, patterns, antipatterns, etc. Sources of information on these additional topics are included later.

Step 2: Create the Initial Team

A small team is usually sufficient when you are starting; you can grow later as needed. You need someone to lead the effort, someone to conduct the SPE studies, and you'll want proper placement in the organization.

Responsibility for SPE

It is important that you designate one or more individuals to be responsible for performance engineering. You are unlikely to be successful without a performance manager who is responsible for:

- Tracking and communication of performance issues
- Establishing a process for identifying and responding to situations that jeopardize the attainment of the performance requirements

- Assisting team members with SPE tasks
- Formulating a risk management plan based on shortfall and activity costs
- Ensuring that SPE tasks are properly conducted

The responsible person should be high enough in the organization to cause changes when they are necessary. The performance engineering manager should report either to the project manager or to that person's manager.

Note: The level of effort required depends on your performance risks. For minor risk projects, one person may be able to support two projects. On high-risk projects you may need several performance engineers. The organizational placement of performance engineers is discussed later.

Finding the Right People

Not everyone is cut out to be a performance engineer. Filling this role requires a special set of talents and interests. A performance engineer is someone who:

- Has diverse experience and interests. Someone who has experience with a variety of software types as well as a quantitative background will be able to pick up the SPE techniques more quickly. A natural curiosity is desirable because it is often necessary to investigate the cause of mysterious performance problems.
- Has good communication skills and a good rapport with others in the organization. A performance engineer needs to work closely with developers, project managers, and corporate leaders. He or she must be able to present results that may be unpopular in a way that fosters communication and cooperation among the various stakeholders to solve the problem.
- Is able to see the big picture. Early in development, SPE requires the ability to distinguish performance-critical areas from the less important parts of the software. A tendency to focus on details rather than on the big picture makes it more difficult to identify those areas.
- Doesn't mind talking to users—and is good at it. Users can provide the best information on how the future system will be used. This helps to identify the Fast Path, workload intensities, execution path probabilities, loop repetitions, and other performance parameters.
- Feels comfortable working with software. Sometimes, especially if you're beginning an SPE effort late in a project, it is necessary to extract details of the software's behavior from the code. It is also necessary to be able to identify viable alternative implementations when the original strategy has unsatisfactory performance.

Performance modeling skills are much easier to develop than these intangible qualities. We have seen several situations where someone who had performance modeling or measurement experience but lacked one or more of the above characteristics was designated as a performance engineer. In each case, there were shortcomings for both for the organization and the individual. For example, no amount of mathematical modeling skill can compensate for the inability to effectively communicate the results, or the inability to identify software alternatives.

Finally, the best candidate is the person who *wants* to do the job. Assigning an uninterested person to the task will result in mediocre SPE efforts. We saw one instance in which the designated person was "unavailable" for key meetings, left work early, and so on. The result was that the person understood only a fraction of what he needed to know, and was unable to adopt and use the SPE models that we had created for the organization.

Proper Organizational Placement

The person responsible for performance engineering should be in the development organization rather than the operations organization. You will have problems if responsibility for SPE is in the operations organization because developers will likely put priority on meeting schedules over making changes to reduce operational costs.

Making SPE a function of the capacity planning group is also a mistake in most organizations, even though that group usually already employs individuals with performance modeling expertise. While some capacity planners have the performance engineering skills listed in the previous section, most are mathematical experts who are too far removed from software issues so it may be difficult for them to identify architecture and design issues and alternatives. If they have a development background and are able to interact effectively with developers, however, this could be a viable option.

Step3: Acquire an Intial Set of Tools

SPE requires tool support for several different tasks: modeling, measurement, development, and reporting. Measurement and reporting tools are generally already available in an organization. You may want to augment them later to make your work easier. Here we focus on modeling tools. You need some tools to conduct the performance analyses. Start small, you can add more later as you gain experience and staff.

Modeling tools for SPE can be categorized as *system modeling* tools or *software modeling* tools. Most system modeling tools support the queueing network model paradigm. Most are intended for use in capacity planning and they are not particularly useful for early SPE studies because they do not focus on the performance characteristics of the *software*. Thus, it is difficult to relate the results produced by the tool to the structure of the software, in order to identify software architectural or design alternatives for solving problems. Using these tools effectively also typically requires a high level of performance modeling expertise. This makes it difficult for developers to use them to create simple models to quickly evaluate design alternatives. They are most useful in later stages of development when it is necessary to model lower-level details of the system, such as a communication protocol.

There are only a few tools specifically oriented to SPE. Those that are available enable modelers to describe the processing steps of the proposed software along with the execution environment, and then evaluate the predicted performance. These tools are better suited to the SPE tasks than system modeling tools, and they do not require special performance modeling expertise for their use. As a result, developers often use them to evaluate their own software.

Tools in this category include *SPE•ED*™ [Smith and Williams 1997], [Smith and Williams 1998], [L&S], and IPS Performance Designer [HyPerformix].

SPE•ED's focus is the software performance model. Users create execution graph models of proposed software processing, and provide specifications of the execution environment. Users can choose to solve either the software execution model or the automatically generated system execution model. The software model is solved analytically; system models may be solved either analytically or by simulation. Model results are presented both with numeric values and color coding, making it easy to identify problematic software processing steps or devices.

IPS Performance Designer uses a spreadsheet-type interface to describe processing tiers and their resource requirements. Models are solved using simulation, and resource consumption and response times are reported. It gives limited information on the design and implementation of individual software components.

Tools such as these simplify SPE evaluations and make it possible for developers to conduct their own studies without needing extensive modeling expertise or assistance from a performance specialist.

Step 4: Strengthen Skills on a Pilot Project

A pilot project is a small-scale project that is conducted to test a process under realistic conditions. It is small scale to keep the costs low. The conditions should be realistic so that you can evaluate whether the process can be used on a full-scale project. A pilot project is often also valuable as a learning tool. Pilot projects are also safer and less expensive than full-scale development efforts for introducing new technologies such as SPE.

Pilot projects are an excellent way to introduce SPE (or any new technology) into your organization. A pilot project allows you to:

- Assess the SPE techniques under realistic, but non-critical, conditions
- Develop standards and refine procedures for use on other projects
- Train key personnel for "seeding" throughout the organization
- Demonstrate the value of SPE to management

Choose a pilot project that is:

- Manageable: A pilot project is just that; it should not be a full-scale development effort. The team should be small (no more than four to six individuals), and the duration of the project should be short (no more than six months).
- Non-trivial: The pilot project should produce results that are visible and clearly valuable to the organization. This means that you should not use a "toy" application. If the results are not clearly relevant to the types of software that you develop, it is too easy to ignore them. The project should also have non-trivial performance issues.
- Non-critical: While the project should be realistic, it should not be one that is critical to your organization's survival or one that is under extreme schedule pressure. If the project gets into trouble, it is too easy to blame the new technology and go back to the old way of doing things without giving it a fair evaluation.
- Measurable: It is important to know whether you've succeeded. While members of the pilot project team may have a good feeling about the project, this is not sufficient to demonstrate success. You should develop objective criteria for evaluating the project, such as: "Does the product meet its performance requirements upon initial completion?" "How precise were early performance model predictions?" "Did performance models identify performance problems that were corrected?" and so on.

Ideally you will be able to get experience on a pilot project first. Sometimes, however, imminent high risk projects dictate that you jump in and learn on the job. It is a good idea to get a mentor to help make sure that your first project is successful.

STEP5: ESTABLISH ON-GOING SPE

Once you have accomplished these tasks, you are ready to establish SPE practices so that they are consistently and systematically applied. You want to officially and formally integrate them into your project planning and development process. SPE deliverables and integration with development processes are covered in [Smith and Williams 2002].

You will likely need a business case for upper management that includes the costs for applying SPE and the savings you expect to achieve.

Business Case

A business case is a document presented to win management commitment for investment in a proposed project or course of action. It establishes that the project will meet an identified business need and is feasible, affordable and a sound investment. If there are competing alternatives, it provides a quantitative basis for choosing among them. The business case also provides a basis for managing the proposed project and measuring its effectiveness.

The concept of preparing a business case to justify a proposed investment is not new. However, in today's economy shrinking budgets, competing proposals for limited funds, and higher fiscal accountability for management have combined to revive the popularity of this tool. Business and government entities from IT departments to human service organizations are now requiring that employees justify new initiatives with a business case.

A business case describes the cash flows (both costs and benefits) that occur as a result of pursuing the proposed course of action and their timing as well as the methods and assumptions that were used in calculating them. It also includes a discussion of critical success factors (e.g., training or the use of consultants), the impact of the project on the organization (will it change the organization chart?), and an identification of any significant risks that could change the outcome along with recommendations for mitigating them.

For example, a business case for SPE would identify the problem to be solved, indicate how SPE can solve the problem, and quantify the costs and benefits of adopting SPE for a given project or the organization as a whole. It would also discuss the impact of SPE on the software development process and identify any risks that might prevent the projected benefits from being realized along with strategies for mitigating them

The essential components of a business case are listed below. The title and format of each section will vary by organization.

Executive Summary. The Executive Summary should be a short summary of your business case; one page is usually best. The rest of the document will provide details to support the summary. This may be the only part of your business case that some people read, however, so you need to make your case here clearly and succinctly. Include a high-level summary of the results and focus on the financial analysis. Leave the details and explanations for the body of the document.

Problem Statement. There is a reason you are proposing this project. For SPE that reason could be a history of performance failures on previous projects or a high risk of failure on a new project. This section should summarize the issues, how they affect the organization and your assessment of what the source of the problem is.

Proposed Solution. This section describes how the problem will be addressed and the expected outcomes. Begin with an overview of the project. Then provide enough detail to demonstrate that what you propose is in line with your organization's business goals and can, in fact, be achieved.

Financial Analysis. The financial analysis details the costs and benefits of the proposed solution and summarizes them using one or more of the financial analysis tools described below. It is based on a *cost model*—a spreadsheet model that includes all of the costs and benefits related to the proposed project.

The model serves as a guide for performing a cost/benefit analysis (see below). The model results are then used to compute financial metrics such as: Return on Investment, Internal Rate of Return, or Total Cost of Ownership. These metrics are discussed below.

Timeline. Each major step in implementing your proposed solution should be shown on a timeline such as a Gantt Chart. These include major milestones (e.g., completion of training) as well as major cash flows (e.g., expenditures such as equipment).

Sensitivity Analysis and Risks. This section discusses potential problems that might prevent achievement of the objectives and overall benefits of the proposal. For example, what if one or more of the assumptions used in the financial analysis is wrong?

Or, what if a step in the process cannot be completed on time?

Sensitivity analysis looks for items in the cost model for which a small change in value can make a difference in the outcome of the analysis. If assumptions were used in deriving these numbers, they should be examined and best- and worst-case estimates used to predict what happens if the assumptions become invalid.

This section should also include any potential risks to the project or organization. For example, if you can't hire a performance analyst by the required date, how will this affect your projected benefits? If these risks can be quantified and used to assign probabilities to model results, this analysis should be included [Schmidt 2003c]. For example, do you have a 50% probability of realizing 100% of your projected benefits and a 90% chance of realizing at least 40% of the projected benefits? For information on risk analysis methods, see [Boehm 1991] and [Boehm 1989]. Also discuss ways of minimizing or mitigating each risk.

Conclusions and Recommendations. This section should summarize the problem, the proposed solution, and the costs and benefits of the solution. Be sure and include information on return on investment or other positive financial outcomes.

It's important to make your conclusions and recommendations explicit. Don't assume that because you have presented all of the evidence your audience will reach the conclusions on their own.

Cost/Benefit Analysis

Cost/benefit analysis weighs the anticipated benefits of a course of action against its expected costs. In performing a cost/benefit analysis, you attempt to quantify every cost and benefit, including seemingly intangible costs or benefits such as reduced employee turnover. If a cost or benefit cannot be quantified, it does not contribute to the financial analysis. That is, it is assigned a value of 0.

Costs. Costs are anything for which you spend money. Examples of costs in an SPE initiative include salaries for performance specialists, tools, and support equipment such as workstations for performance analysts or a dedicated performance testing facility.

Benefits. Benefits are anything that generates revenue or avoids a cost. For SPE, benefits are usually costs due to poor performance that you reduce or avoid as a result of applying SPE. These include: costs of refactoring or tuning, hardware upgrades, contractual penalties, user support costs and others described in the introduction.

Incremental Analysis. Business cases are typically based on incremental cost/benefit analysis. An incremental analysis includes only those costs and benefits that are due specifically to the proposed investment or course of action. Each line item in the financial model includes only changes from "business as usual". For an SPE business case, you would include only costs that are due to adopting SPE (such as software modeling tools) and not costs that would occur whether or not you used SPE. Similarly, you would include only bene-

Cost/Benefit Worksheet			
One-Time Costs	**$**	**Cost Avoidance**	**$**
Tools		Refactoring	$ 812,500
Performance Modeling Tool	$ 8,000	Hardware Upgrade	$ 600,000
Load Driver	$ 70,000	Lost Revenue	$ 975,000
Workstation	$ 4,000	Telephone Agents	$ 325,000
Training			
In-House Training (15 Developers)	$ 66,846		
Performance Engineer	$ 5,923		
Consulting/Mentoring	$ 250,000		
Total One-Time Costs	**$ 404,769**	**Total Cost Avoidance**	**$ 2,712,500**
Recurring Costs (Annual)	**$**	**Intangible Benefits**	
Software Maintenance (Tools)	$ 12,100	Improved Corporate Image	
Salaries (Including Benefits)		Enhanced Customer Relations	
Performance Analyst (1.0 FTE)	$ 100,000	Improved Employee Morale	
Continuing Education	$ 2,200		
Total Recurring Costs	**$ 114,300**		

Figure 2: SPE Cost/Benefit Worksheet

fits that can be directly attributed to SPE (such as avoided refactoring costs).

Figure 2 shows a sample worksheet (adapted from [Reifer, 2002]) for an incremental SPE cost/benefit analysis.The worksheet includes both one-time and recurring costs. One-time costs occur once. They typically include outlays for tools or capital equipment or for project startup costs such as training. Recurring costs are ongoing. They include such things as maintenance on hardware or software licenses or salaries.

"Sunk" Costs. Funds that have already been spent or committed are irrelevant to the analysis and should not be included. These are known as sunk costs. For example, the fact that you held an in-house SPE class three years ago is irrelevant if the development team has turned over completely and everyone needs the training now.

Intangible Benefits. It is important to quantify all benefits. In some cases, this may be difficult. For example, it is difficult to quantify the benefits of reduced employee turnover. However, you can value the *effects* of reduced employee turnover in terms of recruiting expense, training costs, and productivity.

It may be impossible to quantify some benefits. For example, it is difficult to assign a dollar value to "employee morale" and it is likely that your proposed project is only one of many influences that impact employee morale. Benefits such as this are *intangible benefits*.

If you can't reasonably quantify a cost or benefit, it's best to leave it out of the financial analysis. These items are likely to be controversial and leave you open to charges of "padding" the analysis. That does not mean that you can't discuss them elsewhere in the business case. Companies are often willing to invest in "improved customer satisfaction" or "enhanced employee morale" and these important intangibles can tip the scales when the financial analyses for competing alternatives are close.

More information about SPE business cases and a detailed example are in [Williams and Smith 2003].

Other Aspects of SPE

This brief introduction covers only a small part of the SPE techniques. Additional aspects not included here are:

- extensions for modeling Web applications and other distributed systems
- performance walkthroughs and other data gathering techniques
- three key modeling strategies
- software measurement and instrumentation
- performance-oriented design principles
- performance patterns
- performance antipatterns
- late life cycle SPE techniques
- integrating SPE with the software development process

These topics are discussed in [Smith and Williams 2002]. In addition, a method for the Performance Assessment of Software Architectures (PASA^{SM}) is documented in recent papers [Williams and Smith 2002a], [Williams and Smith 2002b]. The Quantitative Scalability Evaluation Method (QSEM^{SM}) is also covered in a recent paper [Williams and Smith 2005]. Other recent publications:

- document new performance antipatterns [Smith and Williams 2002a], [Smith and Williams 2003],
- describe SPE best practices [Smith and Williams 2003b]
- define performance model interchange formats for exchanging models from design tools into modeling tools and among different modeling tools [Smith and Lladó 2004], [Smith, et al. 2005].

These papers are available at www.perfeng.com and through the Computer Measurement Group.

Conclusions

Software performance engineering (SPE) is a method for constructing software systems to satisfy performance requirements. Experience shows that performance of new systems can be orders of magnitude better, with no disruption to delivery schedules, when SPE techniques are systematically applied throughout development.

This paper has presented a five-step approach to establishing SPE in your organization. The steps are:

1. Learn the SPE process, modeling and analysis techniques
2. Create the initial team
3. Acquire an initial set of tools
4. Strengthen skills on a pilot project
5. Establish on-going SPE

An overview of the requirements for each step was presented along with pointers to additional, more detailed information.

References

[Boehm 1991] B. Boehm, “Software Risk Management: Principles and Practice,” IEEE Software, vol. 8, no. 1, pp. 32-41, 1991.

[Boehm 1989] B. W. Boehm, *Software Risk Management*, Washington, IEEE Computer Society Press, 1989.

[CMG 1991] Computer Measurement Group, Software Performance Engineering Panel, moderator C. U. Smith, Computer Measurement Group, December, 1991.

[HyPerformix] HyPerformix, Inc., 4301 West Bank Drive, Building A, Austin, TX 78746, (512) 328-5544, www.hyperformix.com.

[Jacobson, et al. 1999] I. Jacobson, G. Booch, and J. Rumbaugh, *The Unified Software Development Process*, Reading, MA, Addison-Wesley, 1999.

[Kruchten 1999] P. Kruchten, *The Rational Unified Process: An Introduction*, Reading, MA, Addison-Wesley, 1999.

[L&S] L&S Computer Technology, Inc., Performance Engineering Services Division, #110, P. O. Box 9802, Austin, TX 78766, (505) 988-3811, www.perfeng.com.

[Manhardt 1998] D. Manhardt, “Applications Optimization Methodology-An Approach,” *Proceedings of the First International Workshop on Software and Performance*, Santa Fe, NM, October, 1998, pp. 93-100.

[Reifer, 2002] D. J. Reifer, Making the Software Business Case: Improvement by the Numbers, Boston, Addison-Wesley, 2002.

[Rumbaugh, et al. 1999] J. Rumbaugh, I. Jacobson, and G. Booch, *The Unified Modeling Language Reference Manual*, Reading, MA, Addison-Wesley, 1999.

[Schmidt 2003c] M. J. Schmidt, “Business Case Essentials: A Guide to Structure and Content,” Boston, MA, Solution Matrix, Ltd. 2003 (www.solutionmatrix.com).

[Smith, et al. 2005] Smith, C. U., V. Cortellessa, A. Di Marco, C. M. Lladó and L. G. Williams, "From UML models to software performance results: An SPE process based on XML interchange formats", *Proc. 5th Int. Workshop on Software and Performance*, Palma, Illes Balears, Spain, ACM Press, 2005.

[Smith and Lladó 2004] C.U. Smith and C.M. Lladó, An XML-Based Performance Model Interchange Format (PMIF 2.0), *Proc. CMG*, Las Vegas, NV, 2004.

[Smith and Williams 1997] C. U. Smith and L. G. Williams, “Performance Engineering of Object-Oriented Systems with SPE•ED,” *Lecture Notes in Computer Science 1245: Computer Performance Evaluation*, R. Marie et al., ed., Berlin, Germany, Springer, pp. 135-154, 1997.

[Smith and Williams 1998] C. U. Smith and L. G. Williams, “Performance Engineering Evaluation of CORBA-Based Distributed Systems with SPE•ED,” in Lecture Notes in Computer Science, R. Puigjaner, ed., Berlin, Germany, Springer, 1998.

[Smith and Williams 2002] C. U. Smith and L. G. Williams, *Performance Solutions: A Practical Guide to Creating Responsive, Scalable Software*, Boston, MA, Addison-Wesley, 2002.

[Smith and Williams 2002a] C.U. Smith and L. G. Williams, "New Software Performance Antipatterns: More Ways to Shoot Yourself in the Foot", *Proc. CMG*, Reno, NV, 2002.

[Smith and Williams 2003] C.U. Smith and L. G. Williams, "More New Software Performance Antipatterns: Even More Ways to Shoot Yourself in the Foot", *Proc. CMG*, Dallas, TX, 2003.

[Smith and Williams 2003b] Smith, C. U. and L. G. Williams, "Best Practices for Software Performance Engineering", *Proc. CMG*, Dallas, TX, 2003.

[Williams and Smith 2002a] L. G. Williams and C. U. Smith, “PASA^{SM}: A Method for the Performance Assessment of Software Architectures,” *Proceedings of the Third International Workshop on Software and Performance (WOSP2002)*, Rome, Italy, July, 2002, pp. 179-189.

[Williams and Smith 2002b] L. G. Williams and C. U. Smith, “PASA^{SM}: An Architectural Approach to Fixing Software Problems,” *Proc. CMG*, Reno, December, 2002.

[Williams and Smith 2003] Williams, L. G. and C. U. Smith, "Making the Business Case for Software Performance Engineering", *Proc. CMG*, Dallas, TX, 2003.

[Williams and Smith 2005] Williams, L.G. and C.U. Smith, “QSEM^{SM}:Quantitative Scalability Evaluation Method”, *Proc. CMG,* Orlando, FL, 2005.