16

Rightsizing the Cost of Testing: Tips for Executives

Scott Barber

Note to readers: This chapter has been written specifically for senior managers and executives (subsequently, executives) [1]. For the purposes of the discussions in this chapter, an executive is a person who has both the authority and responsibility to allocate the overall budget for software development activities, decide whether testing and testers are owned and managed by projects or are part of a centralized testing service, decide which software will be built or bought, establish or modify project timelines and release dates, or make equivalent decisions in other areas of the corporation such as marketing, product support, or human resources (HR). If this doesn't sound like your role and you don't at least have significant influence over decisions such as these, you are likely to find much of what is contained in this chapter to be beyond your ability to implement. If this is the case, don't feel discouraged or stop reading—the information contained in this chapter can still be immensely valuable to you as you find yourself interacting with executives. Maybe you can even convince your favorite executive to read this chapter and discuss its contents and implications with you as they relate to your corporation. And who knows; maybe one day in the future you'll find yourself in an executive role, dusting off this chapter to help you with a challenge you never imagined you'd be facing when you read it the first time.

16.1 Testing is Just Another operating expense—Mostly

As much as corporate executives may wish that it were not the case, testing will simply never be anything other than an operating expense (OPEX) unless their corporation is in the business of providing testing services as a revenue stream. No matter what fancy accounting is applied to try to make testing appear less costly or what new age calculation is used to justify the cost of testing in terms of return on investment (ROI), for most companies there is simply no direct revenue derived from this expense. For all that it can be painful to admit, testing in most cases is just as much an OPEX as facilities rental, human resources, and the "free" coffee provided to employees in the hopes that it will make them more productive.

Naturally, everyone would notice very quickly if someone were to stop paying the rent on the office, eliminate HR, or sell the coffee pots in an attempt to make a company's bottom line look better as a particularly difficult quarter is coming to a close. In fact, doing any one of those things would probably lead to a near immediate revolt, followed by virtually every employee's resume showing up on every job board in the area. Eliminating these OPEXs under the banner of saving the bottom line would be tantamount to drilling a hole in the hull of a sinking ship while proclaiming that this would help the flood waters drain.

To illustrate, let's take a look at how this worked out for Life Insurance For Testers, Inc. (LIFT). One day, after a particularly bad quarter, the entire functional testing team for LIFT's online policy application product was simply let go [2]. There were a few questions, and some other members of the software development team were nervous for the next few weeks, but there was no revolt among the remaining employees; there wasn't even any whining. Business basically went on as usual. Some of the remaining members of the team spent a little more effort checking their work, but they felt like they had more time to do so, since they weren't being bogged down in dealing with all of those bug reports, "ridiculous demands" from the testers, and incessant questions from product managers about why the testers were finding so many bugs in the first place. In fact, development seemed to be a smoother, happier, faster process—at least for a while.

But then, shortly after one of LIFT's periodic feature upgrade releases, the number of fielded support calls went up dramatically. Before the team could get a patch fix coded and released, LIFT started receiving negative reviews in consumer reports online and in print, and both the new customer policy rate and the term policy renewal rate dropped significantly [3]. There still wasn't a mass revolt, but by the close of the following quarter, LIFT's bottom line was worse than it had been before the testers were let go.

In case you may be wondering, this isn't some dark fairytale I've created to make a point. LIFT itself is a fictitious company, but the LIFT scenarios in this chapter are markedly similar to situations I have firsthand experience with, either as an employee or as a consultant called in to help plug the hole in the hull and bail out the flood waters for companies who have made decisions like these. Over the course of my career, I've witnessed and participated in both failed and successful rescue attempts related to testing cost reduction attempts gone awry. Each situation has been different in significant ways, but one thing was the same every time: when the executives were faced with the fact that their boat was taking on water faster than it was before they trimmed or cut the testing OPEX, they were always initially baffled by how that decision could have had such widespread negative consequences. After all, what executive would knowingly make a decision that's doomed to leave the corporation worse off than it was before and then stick around to watch it happen?

One of the reasons that situations like this occur, and, to varying degrees, occur frequently is that testing is different from most other OPEXs in several fundamental ways:

1. Much more so than facilities, HR, and the presence or absence of available coffee, good testing is often almost entirely transparent at the executive level. It's only when the product is of unacceptable quality that testing is likely to be brought to the attention of executives. After all, executives do tend to have offices, make use of HR services, and drink coffee. But few executives (chief technology officers who got their start as developers could be an exception) make use of a corporation's software testing services very often, at least if things are running as they should.
2. It also takes a relatively long time for too little, poor, or entirely absent testing to cause a problem that is noticeable at the executive level. Unlike when a company switches to a less expensive brand of coffee, which will be noticed within minutes of the first people arriving at work the next day, switching to a less expensive brand of testing might take several release cycles to attract executive attention.
3. From the outside, it is nearly impossible to tell if a software product is good as a result of a good testing program or if it's good because of great development (in spite of poor testing). The converse is also true; from the outside it is very difficult to tell if a product is of poor quality due to poor development, poor testing, a combination of the two, or some other problem such as the product being poorly conceived. True, it is sometimes also difficult to tell if coffee is bad because of

bad beans or bad water, but that is a problem far less complicated (and less expensive) to solve.

4. Even when companies must consider scaling back on facilities, HR programs, or coffee, the reasons these items were valued in the first place are rarely questioned. The question asked when those decisions are made is almost always a form of the following: "Is the added expense of such high quality facilities, services, etcetera, still justified for the incrementally higher value it adds, compared to less expensive options in the face of [insert current situation here]?" But when software product development expenses seem to be outweighing the perceived value obtained from those expenses, one of the questions that is frequently asked instead is "Why are we spending all that money on testing anyway?"

For an executive, all four of these items are worth noting. It's easy enough to avoid the challenges associated with the first three items if executives make some time to get to know their testing program. I'd go so far as to recommend that executives go to the test lab every now and again to lend a hand with the testing for a couple of hours—mostly because I can think of nothing that will increase familiarity with the product more than to spend some time testing it but also because this will give hands-on executives a feel for what their testers do and what challenges they face. Who knows, the visiting executive might even become aware of a challenge that is particularly debilitating to the test program but can be resolved easily through executive channels to increase cooperation and productivity all around.

But whether or not an executive has the opportunity to do some testing of his or her own before deciding whether or not testing costs need to be reduced, the decision must be based on what is being spent on testing vs. the value the company is receiving from that investment.

16.2 Executive Tip 1: Change The Question

If you are an executive who is serious about rightsizing the cost of testing within your corporation, the first thing you need to do is invert this question [4]. Rather than "What value are we currently realizing as a result of what we are spending on testing?" the more useful questions to ask are

"What value do we *want to be* realizing through our testing program, and how much are we *willing to* invest to realize that value?"

Unlike facilities, HR, and coffee, it is at best very difficult to calculate the ROI of a testing program [5]. In fact, it's much like calculating the ROI of auto insurance. The ROI of auto insurance is a big negative number if you own your car outright, live somewhere that you won't go to jail for being an uninsured driver, and never have a claim-worthy incident, but all it takes is one stray roofing nail to puncture a tire on the big rig that you happen to be ever so carefully passing and suddenly that big negative number becomes a big positive number. So which number are you going to use for your ROI calculations? I submit that, absent the development of a highly reliable testing actuarial table, that entire line of thinking will occasionally produce a number that makes sense in retrospect, typically by luck, but most of the time it will simply lead a corporation down a path of gross over- or underspending on testing programs without anyone ever correlating the degree of spending with realized value.

Instead of counting on Lady Luck to smile on you or burying your head in the sand until the testing actuarial table appears, it seems a far better choice to start the rightsizing process by identifying the desired business-level value propositions for your testing program, prioritizing them, and then assigning an acceptable cost to the highest priority value propositions [6].

16.3 Executive Tip 2: Focus on Value To The Business

Identifying the desired value proposition or outcome [7], of a testing program probably sounds like either a redundant task or a complete waste of time, but neither is likely to be the case. Very few testing programs are conceived and maintained with a focus on value to the business. Most testing programs are built on or gradually shift to a focus on project-level or tactical value. As an executive, you certainly don't want to minimize the importance of tactical value, but at the end of the quarter you still need to be able to justify the dollars spent on testing to superiors, the board of directors, investors, and shareholders. If your justification starts and ends with "Testing is just part of how we develop software products," you'll likely be told to bundle it in with the rest of the project costs and let the project managers figure out for themselves how to pay for testing without reducing the product's bottom line [8]. At least, that's what I've most frequently seen done in that situation.

But the truth is that testing does, or at least should, provide business-level value that can be monetized. Here are some examples of how:

- Testing can be the key to preparing for and passing regulatory audits, thus reducing both preparation costs and risk of having to unexpectedly invest in corrective action to pass a potentially more rigorous reaudit.
- Testing can be a very strong defense against claims of negligence, faulty advertising, and service level agreement (SLA) violations (of course, it can also be a very weak defense if the testing wasn't done with this in mind), thus reducing the likelihood of legal action being taken against the company and reducing preparation costs should a suit go to court.
- Testing can help prepare support center staff to field questions about changes and known issues in the product prior to release [9], thus reducing call duration and callbacks while increasing customer satisfaction ratings of support calls immediately following release.
- Testing can provide you with relative quality and stability comparisons from release to release, thus reducing the likelihood of product reviews calling out a downward trend.
- Testing can provide you with necessary information for assessing the relative risk of releasing on schedule versus delaying a release to improve the product, thus allowing you to make an informed decision about which course of action will be less costly overall.
- Testing can provide information about weaknesses in the product, thus enabling you to develop risk mitigation strategies or make other executive-level decisions regarding the product or project.
- Testing can provide start-point data and cross-validation for capacity planning models, thus increasing the accuracy of the models and reducing the likelihood of either overspending or running into unexpected capacity limits.
- Testing can identify candidate builds for prerelease sales demos, along with training materials for sales staff, thus preventing the sales staff from stumbling upon defects and tarnishing the product's credibility in front of potential buyers.

Note that only you and your staff can determine the strategic value of benefits such as these to the business, and that it will be up to your senior staff, generally the beneficiaries of the value, to monetize that value. Also note that these strategic value propositions are not costly add-ons, but are all things that a testing program can provide to the business while already involved in the processes of providing tactical value in such ways as:

- Finding discrepancies between requirements or specifications and what is being delivered, thus reducing the chances of delivering a product that ends up needing to be rebuilt immediately before or shortly after release due to an oversight or misinterpretation
- Detecting issues (bugs) in the product in time to resolve them prior to release when it is cheaper to fix the software or otherwise correct the issue
- Freeing up developers to spend more time focusing on new development and issue resolution and less time focusing on issue detection, thus shortening development cycles and potentially reducing staffing needs, improving the quality of the product, and improving staff contentment
- Serving as the end user's representative to ensure that usability concerns and priority user issues are discovered in time to be addressed appropriately, thus reducing the risk of widespread user complaints, bad press, reduced sales, returns, and so forth due to incorrect assumptions about what matters to the user
- Assisting in determining the scope and impact of detected issues, thus reducing the time needed to decide on a course of action, and decreasing the likelihood that such decisions will be challenged, both at the time and in subsequent scenarios when unexpected problems might crop up

Of course, these are not all or even most of the potential value propositions a testing program can provide; they are just a small sampling. In fact, many of these particulars might not even be all that interesting to your organization. The important part to recognize here is that testing provides both strategic and tactical value propositions and once identified, they can then be prioritized and given a monetized value that we'll refer to as the target cost of the value proposition [10].

Once you've identified, prioritized, and determined a target cost for the desired values, you can quickly estimate whether you have a financially balanced testing program by subtracting the current cost of your testing program from the target cost of the benefits already being realized.

Naturally, you could do this by applying a weighting scheme to the target costs to account for the degree to which their associated benefits are currently being achieved; sum the target costs of what you want, then subtract the cost of what you have to get an estimate of how much you have available to spend to add additional value propositions while remaining balanced; or applying any of a number of other kinds of analysis you favor [11]. The key is recognizing that the calculable value is there in the first place.

16.4 Executive Tip 3: Distribute Testing Costs Carefully

Many organizations assign the costs associated with testing to a single ledger account, typically either the general and administrative (G&A) account or the account that includes the product group or department that "owns" software development. Either of these assignments, or an entirely different one, may be appropriate, so applying a cost allocation model will help determine what cost assignments will really work best for your organization [12].

If you are not familiar with management accounting or cost allocation, here is a quick overview to illustrate how this tip can work in action. First, an account is a subset of a corporation's financial records. Accounts generally map to functional areas or business units of the corporation for the purposes of simplifying financial tracking and accountability of these areas. For our purposes, we'll refer to those functional areas or business units as divisions. Generally speaking, assets, liabilities, equity, expenses, and revenue are tracked within each account.

To illustrate, let's return to LIFT. Needless to say, the eventual fallout from the dismissal of the online policy application's functional testing team attracted executive level attention. As a result of its executives' investigation into how this had happened and how to keep it from happening again, LIFT determined that some organizational changes were necessary to tie financial accountability to the responsibility for delivering value across its testing program.

LIFT tracks its finances across five divisions: G&A, IT, sales & support, product development, and compliance and legal defense. LIFT's desired value propositions for its testing program are the same as the ones discussed earlier. And, as it turns out, each of LIFT's divisions is the primary beneficiary of at least one of the desired value propositions, so LIFT executives have decided to assign the target cost for new propositions and the actual cost for propositions that are already in place to the division that each proposition primarily benefits. The finalized division assignments for LIFT's propositions follow:

G&A—Provide release-to-release comparisons, provide release readiness information, identify areas of weakness to feed risk-mitigation plans

IT—Provide input data and cross-validation for capacity planning

Sales and support—Identify candidate builds for prerelease sales demos and train sales staff, assist in preparing support center for pending release

Compliance and legal defense—Prepare for compliance audits, provide support for legal defense

Product development: All value propositions from the project-level values list

LIFT chose to allocate costs in this manner as a way to ensure that the executives in charge of each of these divisions was clear about what portion of the testing program their divisions were responsible to fund, and these executives could be sure that their divisions were receiving the desired value as a direct result of their investment. Under this organizational model, if a division wants more value from the testing program, then that division knows that it will need to foot the bill. If the division determines that the service isn't as valuable as what it is paying, it can choose to pay for less and get less in return. This also means that since the product development division only funds a portion of the testing program, this division can only eliminate the portion of the program it is funding, thus mitigating the risk of the entire testing function for a product being eliminated without the involvement of executives from all of the affected divisions. Again, as an executive, this is a line of thinking I suspect you are familiar with [13].

16.5 Executive Tip 4: Demand Accountability From Managers of Testing Programs

Things get a little more complicated when one or more of the divisions determine that they want more value but shouldn't have to pay for it or determine that they aren't receiving value equivalent to what they are paying for. There are three primary reasons complications arise when testing programs aren't delivering the desired strategic value:

1. Most executives have approximately zero experience testing software.
2. Most test managers have approximately zero training or experience in business management.
3. Test managers are used to being held accountable for the accuracy, not the value, of the information they provide.

It is common for test teams to have a single-minded focus on identifying defects and championing to have those defects resolved. When asked to provide executive summaries of the collective strategic business implications of those individual defects, testers widely resist [14]. This resistance is generally the result of a confluence of experiences and feelings, the most relevant and common being:

- They recognize that they don't know what risk mitigation measures or fail-safes are in place in production that they have missed in their analysis.
- Previous attempts they may have made to provide this type of information anyway have often been dismissed in an embarrassing or frustrating fashion.
- They feel these additional tasks take time away from their primary mission.
- The request simply doesn't make sense to them.

The first step in resolving this situation is two-way education. Executives need to educate testers, not only about what information they want and how they want it presented, but also about why they want the information and what decisions or actions will result from their having it. Testers need to educate executives about what information is reasonably obtainable and what that information does and does not mean. Only after this cross-education takes place can the test team work productively with executives to determine what information they can exchange to address the company's concerns.

For example, LIFT's executives liked to use statistics based on tests or test cases, and expected such measurements as the number of tests planned, the number executed, and the number of passes and fails to assess the current goodness of the product and the current degree of completeness of testing [15]. This seemed completely reasonable until LIFT recognized that neither test nor test case is a static unit of measure. *Test*, in testing terms, is a less fancy word for "experiment," and *test case* is a fancy way to say "container for one or more tests." After coming to this realization, LIFT executives quickly learned that the number of tests or test cases being planned for each release was really only directly related to the number the test team believed it could accomplish given the parameters of the project. So completing or not completing all of the planned tests was actually an indicator of the test team's estimation skill, not an indicator of whether the desired or necessary testing has been accomplished. What LIFT executives came to understand was that the only way a test group could plan to conduct exactly the right number of the right tests to expose all of the defects that matter would be if they knew what all the defects were and how to find them before they started planning.

LIFT executives came to this realization without even considering that counting tests or test cases doesn't account for the fact that many items that reduce the quality of software products do not lend themselves to binary pass–fail characterization nor the fact that the one particular failing test could be more strategically critical than the combined failure of all of the others. LIFT executives did come to realize, though, that statistical analysis of numbers of tests or test cases can reveal important trends, but goodness and completeness won't be among them. They also learned that achieving the desired value from their testing program involved collaboration in determining what value was desired, what measures or metrics were indicators for that value, and how and when to present those measures and metrics.

It is not necessary for every executive to be aware of this particular metrics disconnect, let alone to be aware of all the other measurement and metrics challenges related to software testing. However, it is critical that the executives who rely on information obtained from a testing program are aware that such challenges do exist and are common. With this awareness, instead of asking for a particular measure or metric, informed executives will be far more likely to engage in a discussion about what value they are counting on receiving from the testing program and collaborating with testers and test managers to devise a measurement or metric that provides that value.

For this to work, the most successful executives will take an approach like the following:

- Kick off discussions with requests like the following: “I need some kind of indicator of [area of interest]. What do you recommend?”
- Be open to learning and collaborating.
- Ensure that the managers of the testing program know that they will be held accountable for providing whatever measure or metric they agree to accurately and in a manner that delivers the desired value—at least if they want to continue receiving funding from the division that the participating executive represents.
- Help testers and test managers understand that they will likely need to change or enhance what they are doing to collect the data that feeds those agreed-upon measures and metrics.

In short, managers of the testing program will need to learn to be accountable to the executives in charge of the divisions that are funding the testing program to ensure that those executives are getting appropriate value for their investment, and those divisions’ executives will need to learn how to work with the managers of the testing program to design the appropriate measures and metrics to be used to deliver that value.

16.6 Executive Tip 5: Keep Tactics at a Tactical level

To this point in this chapter, we’ve not considered the possibility that the testing program might turn out to be simply unable to provide the desired value with the funding available, or using its current methods, tools, or techniques. Nor have we considered what might happen when the testing program is faced with the reality that its income is being reduced while the demand for produced value is on the rise. This is where the other chapters of this book come in.

In today’s climate especially, we are all certainly familiar with the desire for increased efficiency: the appeal—if not the outright necessity—of doing more with less. Every executive who has not yet, at least once, called a middle or line manager into his or her office to tell that person that he or she needs to figure out a way to cut costs in his or her department without degrading the service it provides or delaying project completion, will surely find himself or herself in exactly that position soon. In my experience, this is a fairly common occurrence, except when it comes to test programs. Every time I have encountered cost-reducing measures in a testing program, I have found that those cost-reducing measures to have been decided upon and implemented by an executive more or less independently of interaction with any members of the testing team involved. Whether this happens because of poor lines of communication between departments, a lack of understanding on the part of decision-makers of what testers' contributions to such conversations could be, or a company's sense that testing, like free coffee, is just a necessary luxury whose corners are convenient to cut, my experience strongly suggests that the approach doesn't work out very well.

Part of being accountable, across departments and administrative levels, is being able to apply cost-cutting measures when necessary. Like everyone else, the managers of your testing program need to be held accountable for their program. These managers may need some assistance and training before it's reasonable to hold them solely and independently accountable for big-picture cost-reduction measures, but they are the only ones who can determine what costs can be reduced without significantly degrading the value of their testing program. To use your testing program to its full potential as a source of actual value in your company, it will be necessary to enable the testing team to participate in making these kinds of decisions. In fact, if they aren't the people who put this book in your hands in the first place, you would probably be well served to put copies into their hands so they can consider the cost reduction and cost optimization measures presented in the other chapters as a first step in providing more comprehensive value to your company while doing the work they already do well. Ultimately, the executive who provides his or her testing team with whatever other support and resources team members need to be successful will soon be able to step back and let them prove to the whole company that they deserve the managerial position they occupy.

16.7 Summary

These tips are the result of applying a systems thinking approach to some of the most serious and difficult costing, value, and accountability challenges of managing testing programs that I have encountered over the course of my career. These particular tips draw inspiration from the balanced scorecard approach to corporate performance management, total cost management (TCM), activity-based management (ABM), transfer pricing, target costing, and cost allocation, and have evolved over several years, many clients, significant trial and error, and countless hours of exacting peer review. My goal when writing this chapter was to give executives with at least some degree of responsibility for a software testing program suggestions for how to right size their programs by balancing the programs' costs with the value they provide.

I do not expect that these tips will magically solve all of your company's challenges related to testing costs versus value, but I do believe the principles on which they are based are sound and worthy of your consideration. There are as many possible solutions to an individual company's challenges as there are companies who wrestle with them, and there is not now, nor will there ever be, a one-size-fits-all model for balancing software testing costs versus value. I do believe, however, that to find the right solution for your company, a business-level, systems thinking approach (or an alternate method applied at the same level with equivalent scope) is necessary.

I have shared my thought process, problem-solving approach, and key references in this chapter as one example knowing that you will have no choice but to do your own problem solving to design a solution that is appropriate for the corporation you serve. But I hope I have been successful at inspiring you to design a successful solution of your own and to bring your executive voice into a collaborative exchange with your testers and test managers. Such an approach will be sure to lead to realistic, mutually beneficial value increases for your company as a whole.

Notes

1. By senior managers and executives (subsequently, executives), I'm referring to people who have titles like director of X, vice president of Y, chief Z officer, managing director, and president. Some companies will have additional titles for peers to these positions on the organizational chart. For example, technology companies may have a chief X architect. For the purposes of this chapter, these actual titles are not important. What is important is to note that within this chapter, executives are considered to be individuals within a corporation who have primarily strategic roles, as opposed to line managers or team leads who have primarily tactical roles.
2. Depending on what industry you work on, this group might also be known as quality assurance (QA), system test, or even business analysts. The title isn't important to the illustration. What is important to the illustration is that I am referring to whatever group is primarily responsible for finding bugs and submitting bug reports.

3. If your company develops software that is only used internally, your users may not have the ability or influence to choose an alternate solution, but they certainly can, and will, make it clear that they wish they could. In this case, it is those internal users who will complain—and who may eventually lead an actual revolt.
4. Changing the question is my variation on the systems thinking principle known as formulating the mess. See R. L. Ackoff, *Creating the Corporate Future* (New York: John Wiley & Sons, 1981).
5. It is less difficult but still far from mechanical to calculate the ROI of changing or upgrading a particular aspect of a testing program.
6. The idea of monetizing benefits that have no direct financial value comes from an extension of traditional benefit cost analysis known as social return on investment (SROI). See Scholten, Nicholls, Olsen, and Galimidi, *SROI: A Guide to Social Return on Investment* (Amsterdam: Lenthe Publishers, 2006).
7. Starting with the desired outcome and working back to where things are today, as used here, is an application of interactive planning, specifically the idealization phase, as documented most succinctly in Russell L. Ackoff, "A Brief Guide to Interactive Planning and Idealized Design" (May 31, 2001).
8. An example of throughput costing.
9. Some contributions for which testers are highly valued as part of support center preparation include writing or contributing to FAQs, sharing known workarounds, and having a tester switch places with a support representative during beta release to provide knowledge transfer, cross-training, and a voice for the support center during this last chance to raise and fix issues prior to release.
10. This is actually a hybrid of activity-based costing and target costing applied to a service model as opposed to a product model.
11. This process amounts to applying your favorite performance management method. If you aren't currently using a Performance management method or your current performance management method doesn't handle this scenario particularly well, you might consider balanced-scorecard-derived or -inspired approaches, which I've found to be particularly effective.
12. Tip 3 applies the management accounting concepts of transfer pricing and cost allocation, and presumes that the desired outcome is a purposeful, open, multiminded system with a divisional structure. See Jamshid Gharajedaghi, *Systems Thinking: Managing Chaos and Complexity; A Platform for Designing Business Architecture* (2nd ed., Burlington, MA: Butterworth-Heinemann, 2005).
13. This is a representation of interactive management within a purposeful, multiminded system, as described by Gharajedaghi (2005).
14. Few testers will resist presenting business implications, user implications, support implications, and so forth for individual or closely related defects. Here, I am referring to collective implications of the current state of the product as a whole.
15. For more information about the challenges of test-case-based metrics, see Cem Kaner, James Bach, and Bret Pettichord, *Lessons Learned in Software Testing* (New York: Wiley, 2001). For more information on measurements and metrics, see Jonathan G. Koomey, *Turning Numbers into Knowledge,* (2nd ed., Oakland, CA: Analytics Press, 2008).