



MANCHESTER , UK
21 - 24 November 2011

An excerpt from:
**The Art of Application
Performance Testing**



Ian Molyneaux

Performance testing is a neglected cousin of functional and regression testing, which are well understood in most businesses and where the maturity level is high in many organizations. Curiously, executives lack an appreciation of the importance of performance testing. This has changed very slowly over the past ten years, despite the best efforts of consultants like myself and the many highly publicized failures of key software applications.

Contents

Why Performance Testing	2
What is Performance? The End User Perspective	2
Bad Performance: Why It's So Common	6
Summary	11
Biography	11

This eBook is the first chapter from
‘The Art of Application Performance Testing’
by Ian Molyneaux.

Why Performance Testing?

An abstract from: “Faster than a speeding bullet... Superman, Action Comics”

This chapter poses some fundamental questions about the subject of this book. What is performance? Why carry out performance testing in the first place? Here I also strive to provide a definition of when an application is considered performant and when it is considered nonperformant, and then discuss some of the common causes of a less than perfect end user experience.

Non-performant (i.e., badly performing) applications generally don’t deliver their intended benefit to the organization. That is, they create a net cost of time, money, and loss of kudos from the application users, and therefore can’t be considered reliable assets. If an application is not delivering to the business, its continued existence is definitely on shaky ground, not to mention that of the architects, designers, coders, and testers (hopefully there were some!).

Performance testing is a neglected cousin of functional and regression testing, which are well understood in most businesses and where the maturity level is high in many organizations. Curiously, executives lack an appreciation of the importance of performance testing. This has changed very slowly over the past ten years, despite the best efforts of consultants like myself and the many highly publicized failures of key software applications.



— TWEETABLE —

If an end user perceives bad performance from your website, their next click will likely be on your-competition.com

What is Performance? The End User Perspective

When is an application considered to be performing well?

My years of working with customers and performance teams suggest that ultimately, the answer is one of perception. A well-performing application is one that lets the end user carry out a given task without undue perceived delay or irritation. Performance really is in the eye of the beholder.

With a performant application, users are never greeted with a blank screen during login, and can achieve what they set out to accomplish without letting their attention wander. Casual visitors browsing a Web site can find what they are looking for and purchase it without experiencing too much frustration, and the call center manager is not being harassed by complaints of poor performance from his or her operatives.

It sounds simple enough, and you may have your own thoughts on what constitutes good performance. But however you define it, many applications struggle to deliver an acceptable level of performance by this definition.

Of course, when I talk about an application I’m actually referring to the sum of the whole, as an application is made up of many components. At a high level we can define these as the application software plus the application landscape. The latter includes the servers that are required to run the application software along with the network infrastructure that allows all the application components to communicate.

If any of these areas has problems, application performance is likely to suffer.

You might think that all we have to do to ensure good application performance is to observe the behaviour of each of these areas under load and stress and correct any problems that occur. The reality is very different in that this is often too little too late and you end up dealing with the symptoms of performance problems rather than the cause.



TWEETABLE

A well-performing application is one that lets the end user carry out a given task without undue perceived delay or irritation. Performance really is in the eye of the beholder.

Performance Measurement

So how do we go about measuring performance? We've discussed end user perception, but in order to accurately measure performance there are a number of key indicators that we have to take into account. These will form part of performance requirements that I will discuss further in Chapter 2 but for now we can divide them into two types: service-oriented and efficiency-oriented.

Service-oriented indicators are availability and response time and refer to how well (or not) an application is providing a service to the end users. Efficiency-oriented indicators are throughput and utilization and focus on how well (or not) an application makes use of the application landscape. Briefly we can define them as follows:

Availability

The amount of time an application is available to the end user. Lack of availability can be

very significant, as many applications will have a substantial business cost for even a small outage. In performance testing terms, this refers to the complete inability of an end user to make effective use of the application.

Response time

The amount of time it takes for the application to respond to a user request. For performance testing, one normally measures system response time, which is the time between the user requesting a response from the application and a complete reply arriving at the user's workstation.

Throughput

The rate at which application-oriented events occur. A good example would be the number of hits on a web page within a given period of time.

Utilization

The percentage of the theoretical capacity of a resource that is being used. Examples include how much network bandwidth is being consumed by application traffic, and the amount of memory used on a server when one thousand visitors are active.

Taken together, these indicators can provide us with an accurate idea of how an application is performing and its impact in capacity terms on the application landscape.

Performance Standards

By the way, if you were hoping I could point you to a generic industry standard for good and bad performance, you're out of luck because no such guide exists. There have been various informal attempts to define a standard, particularly for browser-based applications—so for instance, you may have heard the term “minimum page refresh time.” I can remember a figure of 20 seconds being bandied about, which rapidly became

8 seconds. Of course, the application user (and the business) wants “instant response” (in the words of the Eagles band, “Everything all the time”), but this sort of performance is likely to remain elusive.

Many commercial Service Level Agreements (SLA's) relate to infrastructure performance rather than the application itself, and often only to a specific area such as network latency or server availability.

The following list summarizes research conducted in the late 1980s (Martin 1988) that attempted to map user productivity to response time. The original research was based largely on green-screen, text applications, but I suspect its conclusions are still very relevant.

Greater than 15 seconds

This rules out conversational interaction. For certain types of applications, certain types of users may be content to sit at a terminal for more than 15 seconds waiting for the answer to a single simple inquiry. However, to the busy call centre operator or futures trader, delays of more than 15 seconds may seem intolerable. If such delays can occur, the system should be designed so that the user can turn to other activities and request the response at some later time.

Greater than 4 seconds

These delays are generally too long for a conversation requiring the operator to retain information in short-term memory (the operator's memory, not the computer's!). Such delays would be very inhibiting in problem-solving activity and frustrating in data-entry. However, after the completion of a transaction, delays of 4 to 15 seconds can be tolerated.

2 to 4 seconds

A delay longer than 2 seconds can be inhibiting to operations that demand a high level of concentration. A wait of

2 to 4 seconds at a terminal can seem surprisingly long when the user is absorbed and emotionally committed to complete what he or she is doing. Again, a delay in this range may be acceptable after a minor closure. It may be acceptable to make a purchaser wait 2 to 4 seconds after typing in her address and credit card number, but not at an earlier stage when she is comparing various product features.

Less than 2 seconds

When the application user has to remember information throughout several responses, the response time must be short. The more detailed the information remembered, the greater the need for responses of less than 2 seconds. Thus, for complex activities such as browsing camera products that vary along multiple dimensions, 2 seconds represents an important response-time limit.

Subsecond response time

Certain types of thought-intensive work (such as writing a book!), especially with applications rich in graphics, require very short response times to maintain the users' interest and attention for long periods of time. An artist dragging an image to another place in a document he's formatting needs to be able to act instantly on his next creative thought.



TWEETABLE

Performance testing is a neglected cousin of functional and regression testing, which are well understood in most businesses and where the maturity level is high in many organisations.

Deci-second response time

A response to pressing a key and seeing the character displayed on the screen or clicking a screen object with a mouse needs to be almost instantaneous: less than 0.1 second after the action. Many computer games require extremely fast interaction.

As you can see, the critical response time barrier seems to be 2 seconds. Response times greater than this have a definite impact on productivity for the average user, so our nominal page refresh time of 8 seconds for Internet applications is less than ideal!

rely on cyberspace for a good deal of their revenue in what is probably the most competitive environment imaginable. If an end user perceives bad performance from your website, their next click will likely be on your-competition.com.

Internet facing applications are also the most vulnerable to sudden spikes in demand, as more than a few high profile retail companies have discovered to their cost at peak times of the year such as Christmas and Thanksgiving.

The Internet Effect

The explosive growth of the Internet has contributed in no small way to the need for applications to perform at warp speed. Many (or is that most?) businesses now



TWEETABLE

Developers don't deal with the large number of users who are still at the end of low-bandwidth, high-latency WAN links.

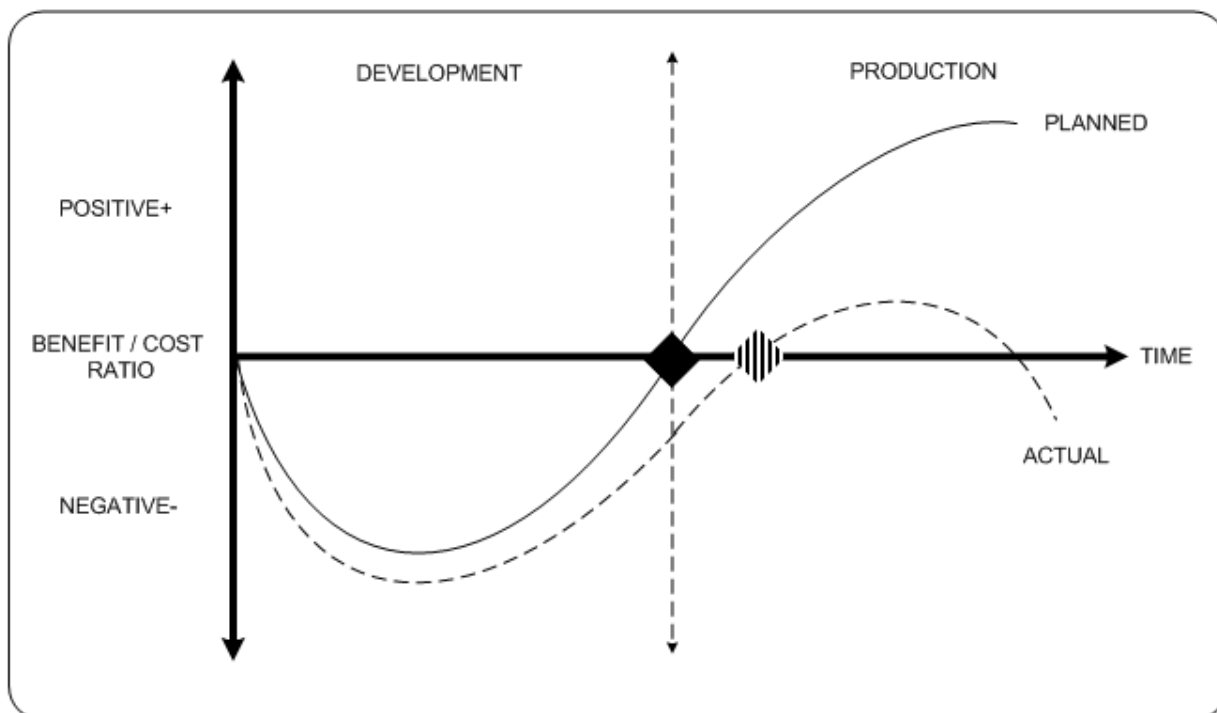


FIGURE 1-1. The IT business value curve

Bad Performance: Why It's So Common

OK, I've tried to provide a basic definition of good and bad performance. It seems obvious, so why do so many applications fail to achieve this noble aspiration? Let's look at some common reasons.

The IT Business Value Curve

Performance problems have a nasty habit of turning up very late in the application life-cycle, and the later that you discover them the greater the cost to resolve. Figure 1-1 attempts to illustrate this point.

The solid line (planned) indicates the expected outcome where the carefully factored process of developing an application comes to fruition at the planned moment (black diamond). The application is deployed successfully on schedule and immediately starts to provide benefit to the business with little or no problems post deployment.

The broken line (actual) demonstrates the all too frequent reality when development and deployment targets slip (striped diamond) and significant time and cost is involved in trying to fix performance issues in production. This is bad news for the business as the application fails to deliver the expected benefit.

Resolving Performance Defects (2006)

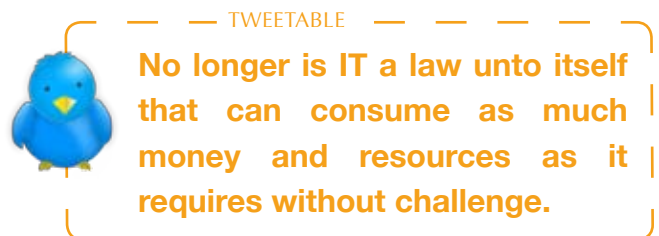
Approach	% Resolved in Production
Firefighting	100%
Performance Validation	30%
Performance Driven	5%

Source: Forrester Research

FIGURE 1-2. Forrester Research on resolution of performance defects

These sort of failures are becoming increasingly visible at the board level as companies seek to implement Information Technology Service Management (ITSM) and Information Technology Portfolio Management (ITPM) strategies on the way to the holy grail of Information

Technology Infrastructure Library (ITIL) compliance. The current frame of reference considers IT as just another (important) business unit that has to operate and deliver within budgetary constraints. No longer is IT a law unto itself that can consume as much money and resource as it requires without challenge.



Performance Testing Maturity - What the Analysts Think

But don't just take my word for it. Figure 1-2 is based on data collected by Forrester Research in 2006 looking at the number of performance defects that have to be fixed in production for a typical application deployment.

As you can see three levels of performance testing maturity were identified. The first one, "Firefighting," occurs in a situation where little or no performance testing was carried out prior to application deployment, so effectively all performance defects have to be resolved in the live environment. This is the least desirable approach, but is surprisingly still relatively common. Companies in this mode are exposing themselves to serious risk.

The second level "Performance Validation (or Verification)," covers companies that set

aside time for performance testing, but late in the application life cycle. So a significant number of performance defects are still found in production (30%). This is where most organisations currently operate.

The final level, “Performance Driven,” is where performance considerations have been taken into account at every stage of the application life cycle. As a result, a very small number of performance defects are discovered post deployment (5%). This is what companies should be aiming for as their performance testing model.

Lack of Performance Considerations in Application Design

Returning to the discussion of common reasons for failure. If you don’t take performance considerations into account during application design, you are asking for trouble. Good design lends itself to good performance, or at the very least the agility to change or reconfigure an application to cope with unexpected performance challenges.

Design-related performance problems often remain undetected until very late in the application life-cycle where, as we’ve discussed they are very difficult to overcome completely, and sometimes impossible without significant application reworking.


Most applications are built from software components that can be tested individually and may perform well in isolation, but it is equally important to consider the application as a whole. These components have to interact in an efficient manner to achieve good performance.

Performance Testing is Left to the Last Minute

As mentioned, most companies are in “Performance Validation (or Verification)” mode, where performance testing is done just before deployment with little consideration for the amount of time required or for the ramifications of failure. This isn’t as bad a situation as “Firefighting.” But it still carries a significant degree of risk that you won’t identify serious performance defects that may appear in production, or allow enough time to correct problems that are identified before deployment.

The all-too-common result is a delay to the application roll-out while the problems are sorted out, or an application that is deployed with significant performance issues requiring costly, time-consuming remedial work after deployment. Even worse, the application may have to be withdrawn from circulation entirely while it’s battered into shape.

All of these outcomes have a very negative affect on the business and on the confidence of those expected to use the application. You need to test for performance issues as early as you can, rather than leave it to the last minute.



Many companies underestimate the popularity of their new web applications. This is partly because they are deployed without taking into account what I call the “novelty factor.”

How Many Users Are There?

Put another way, little thought is often given to capacity or sizing. Developers and testers often overlook the size and geography of the end-user community. I think it would be true

to say that many applications are developed and subsequently tested without more than a passing thought for the following:

- How many end users will actually use the application?
- How many of these users will use it concurrently?
- How will the end users connect to the application?
- How many additional end users will require access to the application over time?
- What will the final application landscape look like in terms of the number and location of the servers?
- What effect will the application have on network capacity?

This oversight manifests itself in unrealistic expectations of the number of concurrent users that the application is expected to support. Furthermore, developers don't deal with the large numbers of users who are still at the end of low-bandwidth, high-latency WAN links. I will cover connectivity issues in more detail in Chapter 2.

Underestimating Your Popularity

This might sound a little strange, but many companies underestimate the popularity of their new web applications. This is partly because they are deployed without taking into account what I call the “novelty factor.” When something's shiny and new, people generally find it interesting, so they turn up in droves. Therefore, the 10,000 hits you had carefully estimated for the first day of deployment suddenly becomes 1000,000 hits and your application infrastructure goes into meltdown!.

Putting it another way, you need to plan for the peaks rather than the troughs.

SPECTACULAR FAILURE: A REAL WORLD EXAMPLE

Some years ago, the UK government decided to make available the results of the 1901 census on the Internet. This involved a great deal of effort converting old documents into a modern digital format and creating an application to provide public access.

I was personally looking forward to the launch, as at the time I was tracing my family history and this promised to be a great source of information. The site was launched and I duly logged in. Although I found things a little slow, I was able to carry out my initial searches without too much issue. However when I returned to the site 24 hours later, I was greeted with an apologetic message saying that the site was unavailable. It remained unavailable for many weeks before finally being relaunched.

This is a classic example of underestimating your popularity. The amount of interest in the site was far greater than anticipated, hence it couldn't deal with the volume of hits.

Now, I'm not saying that there wasn't any performance testing carried out prior to launch. But I would suggest that the performance expectations for the site were perhaps a little conservative.

You have to allow for those peaks in demand.



— TWEETABLE —

Searching for performance testing on the internet is more likely to present you with information about sport cars or even washing machines.

Performance Testing is Still an Informal Discipline

As mentioned earlier, performance testing is still very much an informal discipline. The reason for this is hard to fathom, because functional/regression testing has been well established for many years. There is a great deal of literature and expert opinion available and many established companies who specialize in test execution and consulting.

For performance testing, the converse is true, at least in terms of reference material. One of the reasons that I was prompted to put (virtual) pen to paper was the abject lack of anything in the way of written material focused on application performance testing. There are a myriad of publications that explain how to tune and optimize an application, but nothing about how you set about effective performance testing in the first place. If you're a budding performance tester, you have been very much on your own (until now!).

Try it for yourself. Searching for performance testing on the Internet is more likely to present you with information about sports cars or even washing machines.

Not Using Automated Testing Tools

Ninety-nine percent of the time, you can't carry out effective performance testing without using automated test tools. Getting a hundred (disgruntled) staff in at the weekend (even if you buy them all lunch) and strategically deploying people with stop-watches just doesn't work. Why? You'll never be able to repeat the same test twice. Furthermore, making employees work for 24 hours if you find problems is probably a breach of human rights.

Also, how do you possibly correlate response times from 100 separate individuals, not to mention what's happening on the network and the servers? It simply doesn't work unless your application has fewer than five users, in which case you probably don't need this book.

A number of vendors make great automated performance testing tools. Costs will vary greatly depending on the scale of the testing you need to execute, but it's a competitive market and biggest is not always best. So you need to do your homework and prepare a report for your finance department. Appendix C contains a list of the leading vendors.)



TWEETABLE

If you don't take performance considerations into account during application design, you are asking for trouble.

Application Technology Impact

Certain technologies that are commonly used in creating applications didn't work well with

the first and even second generation of automated test tools. This has become a considerably weaker excuse for not doing any performance testing, as the vast majority of applications are now web-enabled to some degree. Web technology is generally well supported by the current crop of automated test solutions.

Development and deployment have crystallized by now onto a (relatively) few core technologies. Accordingly, most automated tool vendors have followed suit with the support that their products provide. I will look at some common application technologies and their impacts on performance testing in Chapter 5.

Summary

This chapter has served as a brief discussion about application performance, both good and bad. I've touched on some of the common reasons that failure to do effective performance testing leads to applications that do not perform well. You could perhaps summarise the majority of these reasons with a single statement:

Testing for performance is not given the importance or consideration that it deserves as part of the application life cycle.

In the next chapter we move on to a discussion of the building blocks that are required to implement an effective application performance testing strategy: requirements.

TWEETABLE



You need to test for performance issues as early as you can, rather than leave it to the last minute.

Biography



Ian Molyneaux has over 30 years' experience in IT, providing performance consultancy and promoting business-driven performance testing. He joined Int Technica in 2010 as Head of Performance; since then he has grown the team to 6 full time members of staff, and has been instrumental in the creation of Int Technica's Performance Assurance Framework. Prior to this role he held senior performance roles for Compuware and MicroFocus. Ian's book "The Art of Application Performance Testing" was published by O'Reilly in 2009. He wrote the book in response to the lack of any practical guide to the field of performance testing, and whilst it hasn't yet made him a rich man, he has a growing international community of followers and fans.

This eBook is an excerpt from "The Art of Application Performance Testing" by Ian Molyneaux. To buy the full book [click here](#).

Join the conversation...

Access the latest testing news! Our Community strives to provide test professionals with resources that prove beneficial to their day to day roles. If you would like to contribute and have your work published online please contact us! Keep in touch.....



Follow us on **Twitter** @esconfs

Remember to use our hash tag #esconfs when tweeting about EuroSTAR 2011!



Become a fan of EuroSTAR on **Facebook**



Join our **LinkedIn Group**



The EuroSTAR Blog
Written by Testers for Testers

Contribute to the **EuroSTAR Blog**



Check out our free **Webinar Archive**



Download our latest **eBooks**

www.eurostarconferences.com