

# Completely Failing at: Performance Testing



In this article Phil Rough takes a look at performance test projects and lists his **top ten reasons for failure.**

**Phil Rough** is a Principal Consultant for Thorough Consulting, a consultancy dedicated to optimising the performance of critical business systems.

It has always astounded me as to why 'testing' is seen as such a generalist area. Every person that is involved in the various phases of testing is seen as a tester even though each stage requires its own level of knowledge and expertise. A search through the job postings by IT recruiters will explain what I mean.

A recent search for 'performance tester' resulted in pages of posting for database administrators, developers and software testers. Equally, a search for a popular load testing tool resulted in pages of postings requiring experience of functional testing, test management and even user acceptance testing.

Clearly recruitment consultants do not understand the subject of performance testing. But this shouldn't come as such a surprise. During my years in the testing industry I have found that many companies do not 'get' performance testing either.

Why do performance test projects fail to deliver?

I have found that companies understand that performance testing needs to be done; that the project cannot be completed without a tick in the box and that releasing a system that doesn't perform can be costly. High profile performance failures do still occur even after the experiences of the Dot Com era. But it doesn't always follow that a performance test project will get the attention it deserves. Performance testing is not always part of the main project plan. There is often no budget allocated, no time

scheduled and no resources requested. The planning stage is often overlooked with no clear definition of objectives, requirements or deliverables. The execution phase is often rushed, restricted by the availability of the system and the results are maybe inaccurate, often disputed and ultimately meaningless.

In answer to the question, I have listed my top ten reasons for failing at performance testing. The list is based upon the misconceptions of many people that I have heard while engaged in software development projects. For a company that is about to embark on a performance test project and is looking to fail - then I would recommend that all the points are adhered to. However, **my advice to any sensible project would be to follow the exact opposite of the list.**

As any performance tester would agree there is more to a performance testing than meets the eye. The knowledge and experience of a performance tester is different from that of a functional tester, which in turn is different to that of a user acceptance tester. Each is important and each is essential for the completion of a successful testing project ■

[philrough@thoroughconsulting.co.uk](mailto:philrough@thoroughconsulting.co.uk)





**Don't Be Specific**

Testing is about finding problems in the system. The result of a test is a system pass or fail, a green or a red, a 'Go' decision or a 'NotJustYet' decision. So, don't be specific about the test. If you are going to execute a performance test against a system then go ahead and test it for performance and scalability and capacity and reliability and endurance and resilience and simply report it in a single result.



**Don't Check the System**

The architect has spent a long time designing the system. The test environment will be exactly as appears in the architect's diagramme. The configurations might change a little when required but it will be mostly correct. The configuration will probably change when the system goes into production anyway.



**Don't Act on the Results**

Everyone disagrees with the result of the performance test. The development team blames the testers. The administrators disagree with the timings and the architect will question the environment that was tested against. Even if the results are valid the performance will be different in the production system so action won't be required.



**Don't Schedule Time**

The system can not be performance tested until it is ready. So leave performance testing right to the very end of the project. The bit in between the functional testing and user acceptance testing - or maybe even after that. That bit that gets dropped as the project overruns.



**Don't Use Tools**

Test tools are costly. They are expensive to buy, problematic to implement and time-consuming to use. Performance testing can be done manually or, if the worst comes to the worst, the development team can always build something if it is needed. Beer and pizza anyone?



**Don't Write a Plan**

Performance testing doesn't need to be planned. A formal document doesn't need to be created, followed or maintained. The performance tester will know what is required and should not need further resources or scheduling. The result of the test will be evidence of the plan in itself.



**Don't - Performance Test**

Your business critical system uses the best hardware and your development team have used the best software. You have got the best teams of people from developers to DBA's to architects to system administrators. You have selected the best vendor and chosen the best implementation partner. Of course the system will perform. It does everywhere else doesn't it?



**Don't Dedicate an Environment.**

System environments are expensive. The cost of hardware, software and maintenance all add up. The test environment should have the up-to-date versions of code, configuration and hardware components. The performance test itself can always be scheduled out of office hours to lessen the impact.



**Don't Use Experts**

Performance testing is just part of testing. Therefore, one of the business analysts who has been drafted in to work in the test team can be responsible for it. They could be given a book to read or even sent on a training course. After all, performance testing isn't difficult.



**Don't Understand the system under test**

A system is a system is a system. At the end of the day you just write a test and run it. A tester shouldn't need in-depth knowledge of the system - there are developers and administrators for that.

**Subscribe to**  
Professional Tester Magazine