

# The TelQA Testing Series

## Part 2: Performance Testing Aims



## The TelQA Testing Series

This article is part of a series of documents which cover the basics of performance testing. See [www.telqa.com](http://www.telqa.com) for other articles in this series and additional downloadable testing resources.

### Overview

This article provides you with an introduction to determining the aims and objectives of your performance tests.

During the course of this article we will look at the performance testing aims for:

- System testing – performance testing for a complete system/application
- Development phase testing – performance tests that support application development

### System Testing

To begin with, let's first consider performance tests which target a complete system. We can split the performance testing aims into four broad categories:

- Quantifying the overall performance level of a system
- Comparing the performance of a system against known performance thresholds (e.g. Service Level Agreements)
- Assisting with performance tuning or the removal of performance bottlenecks
- Assessing system scalability

Whilst it is possible to achieve all four of these aims with a single set of tests which quantify the full system performance, in practice the time and resources required to design and implement a full system performance analysis are unlikely to be justified when only specific performance related data types are required.

The key point to note is that the aim of the test should be used to determine the type of applied performance test, i.e. the test development process should focus on matching the type of test to the required aims and not try to provide generic performance tests which attempt to quantify the entire system with a limited number of tests.

Where the requirements involve a comparison with known performance thresholds, the scope of the test can be restricted and hence the test development and deployment time can be significantly reduced. However, the results of these types of tests typically provide no indication of the overall system performance.

Detection of performance bottlenecks may involve very specific testing using system loads which are unrepresentative of the expected production environment. Again, this type of test will provide little or no indication of the overall system performance.

System scalability tests are very similar to tests which are designed to quantify the overall system performance, but may use different loads which are not representative of the current production loads but are intended to be representative of future load types.

## Development Phase Testing

The aims of a performance testing during the application development phase can be broadly categorised as:

- Feasibility testing for in-house or third-party hardware and software
- Unit testing
- Integration testing

Performance testing carried out as part of a feasibility test will typically use specific loads. For example, when selecting an appropriate database prior to system development, a performance test would be carried out to determine the ability of the database to provide the required level of performance. This type of test would typically utilise database specific communications which, whilst being representative of the intended use, would not necessarily be the communications used in the finished system.

Unit testing may utilise performance tests to determine whether individual units (or parts) of the application perform to a specified level. Alternatively a performance test may be intended to determine the overall performance of the unit prior to integration testing.

Performance tests are also commonly used as part of integration testing. The aim of this testing is to determine whether the performance of the individual units are affected when one or more units are integrated.

## The TelQA Testing Series

Part 1: The Performance Testing Process

**Part 2: Performance Testing Aims**

Part 3: Types of Performance Testing

Part 4: Performance Test Environments

Part 5: Performance Test Inputs and Outputs

Part 6: Capturing User Interactions

Part 7: Modelling Scripts

Part 8: Selecting Performance Monitors

Part 9: Performance Test Creation

Part 10: Configuring Performance Test Environments

Part 11: Verifying Performance Test Environments

Part 12: Running a Performance Test

Part 13: Analysing Performance Data

Part 14: Using Performance Data for Tuning and Bottleneck Removal