White Paper: The Principles of Capacity Management

Sarquol Limited

© Sarquol Limited 2006.

This White Paper has been provided for the use Sarquol's clients, and of members of Sarquol's web site. The information within it may be used for personal or commercial purposes, and may be copied and distributed in its whole form at will. The paper should not be published or reproduced in part without explicit written permission of Sarquol.

Requests to distribute or copy the White Paper in part should be made to sales@sarquol.com, or in writing to the company at:

1 Bellevue Road,
Whitstable,
Kent. England.
CT5 1PU

Change History

Version	*Date*	*Changes*
1.0	16 Mar. 2006	Initial Issue

Table of Contents

1. Introduction 4
Context 4
Audience 4
2. Project Overview 5
Processes 5
Project Overview 6
Project Lifecycle 7
3. Performance Modelling and Testing 13
Performance Model outline 13
Performance Testing 13
Ideal performance testing 14
Ideal UAT and production environment 16
Ideal Tests 16
4. Sarquol Limited 17

1. Introduction

Context

When creating a system it is common knowledge that its performance is important, since users of the system will be put off it is deemed to be too slow. It is also common knowledge that performance testing tools such as OpenSTA can create load tests to help make sure that your system will perform. The use of these and other tools and techniques to make sure that the system as a whole performs well is, however, rather less well understood. This document outlines an approach to fitting performance testing into a company's development process and so provides a context for managing system performance.

Audience

The document is intended for anyone who needs to understand how to go about verifying the likely performance of a developed system, and to make sure that an appropriate approach is used to manage the performance of the system. The document more describes the management approach required for the Capacity Planning and Management, rather than going into the technical details. As such, it does not assume detail technical knowledge or go into the mathematical basis for performance modelling or testing.

Note: It is planned to produce a more technical papers at a later date which will contain this sort of information. If you need this, or require consultancy in the management of a system's performance than please request this via our web site or by e-mail to sales@sarquol.com.

2. Project Overview

Processes

The main processes involved in capacity management are shown in Figure 1. The "Evaluate Risk" process has been placed in the centre, since it drives the need for the other major processes. To be able to evaluate the risk, however, requires a level of knowledge about the system to be deployed, and its likely usage. It is for this reason that the other processes exist.

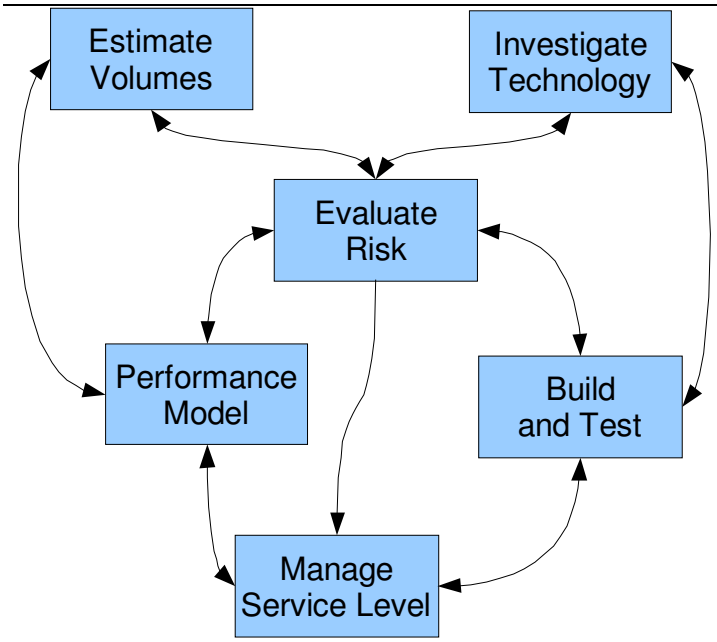


Figure 1: The main processes for Capacity Management

A quick summary of each of the processes is:

Process	*Description*
Estimate Volumes	The performance of any complex system is directly dependent on the load placed on the system, and the volume of data within it. Thus it is necessary to have an idea of the load that will be placed on the system, which should be in terms relevant to the business application of the system. It is important to include in this investigation the variation with time of the system's use. Flat usage profiles just don't happen.
Investigate Technology	To manage capacity it is important to understand what you are managing the capacity of. You need to understand what is under your control and what is not, and to have an idea of how all the components of the system will react at different load levels.
Evaluate Risk	Understand what might go wrong in the system as a whole, and what the impact of this happening would be. Capacity Management, in common with most management activities, is often simply a case of trading off the risks that exist until a

Process	*Description*
	reasonable way forward is decided on.
Performance Model	To manage the performance and risk of the system you need to understand it. You need to be able to understand how variation in the volumes of the system will affect the end users. You need to be able to understand how this can be offset by scaling the size of different components within the system. To do this there are two approaches, do it in reality or model the system and try it out with a model. I would recommend always starting with some form of model.
Build and Test	In the end you will need to have a system and to test the system against the required volumes. The "Build" here should be read as all forms of coding, configuration, tuning and optimisation. It includes activities such as optimising database indices, and adding new hardware. The testing is the application of load testing tools in a development system, as well as production monitoring systems.
Manage Service Level	It is important to understand the level of performance that is considered "necessary and sufficient." Testing, adding hardware and optimisation are important, time consuming and expensive processes. The Service Level for a system should define the agreed volumes over time and the acceptable performance for those volumes. The operational system must then be tested against this agreement on an ongoing basis. The Service level agreement must take account of the statistical nature of performance management. Thus a statement such as “All operations must complete is less than 2 seconds” is insufficient. It could be improved by stating “For all critical operations they must be completed from initiation by the user to the system result being visible in 2s for 95% of cases.” The level of certainty required is as important as the figure chosen. In both cases choosing more stringent criteria is likely to result in more hardware and development costs.

Project Overview

The processes have not been placed in a Project sequence because it is rarely possible to take a waterfall approach to Performance Testing. In the ideal world the diagram would flow neatly from deciding on volumes, to evaluating the technology, and the building and testing the system. The test would, of course, pass first time because everything was already known. There would be no need to build a performance model or evaluate the risk of the system, because everything would already be known.

This is, of course, a fantasy. In a real world project the system to be built isn't yet fully understood, and won't be until it has been constructed. There are only vague ideas of the level of activity that can be expected. We don't have a performance model, and aren't even sure what one should look like. We know there is a lot of risk

because our new customers will go away if the system doesn't perform well enough. Lastly, we have the vendor's pitch that the product we have bought performs really well but we don't have any benchmark figures which seem to be in any way relevant to us.

Project Lifecycle

The following outlines the activities that are most likely to be taking place as the Project Lifecycle progresses. The description is staged, but I have tried to avoid using the stage names from any particular methodology. Hopefully it will be reasonably obvious how these stages fit into your particular methodology.

Feasibility

When a project is first started there is likely to be very little solid information about how everything is likely to move forward. What is likely to be possible is to be able to "put a stake in the ground" and set initial principles and budgets. At this stage consider a range for all estimates: optimistic, planned and pessimistic. Keep everything at fairly high level, and take care to document assumptions as rigorously as possible.

Process	*Activity*
Estimate Volumes	There is likely to be a high level business case, which can start to define the likely volumes that will be needed in order to meet that business case. Keep these as business focussed as possible.
Investigate Technology	Check what sort of solutions may be possible. Use research of RFI submissions that are going out to vendors to find how the various technologies are likely to perform. Find out if you have access to information on performance at present.
Evaluate Risk	As for all risk management, assess what is most likely to go wrong and what the impact of that might be. Use what information you have about the available technologies and volumes to focus the risk consideration.
Performance Model	At this stage only very simple performance modelling is justified. Mostly this will be simply inference from other data, or speculation based on what is known. It should still be considered what can best add value at this stage though.
Build and Test	The high level return on investment considerations in the business case will also provide an initial idea of the budget for creating and running the system that may be available. This will control the sort of build and test that can be planned in. At this stage the effort in this area is likely to be main
Manage Service Level	Even at this stage there are liable to be incongruously accurate statements about the required performance of the system. Record these and add a disclaimer that such values will only be able to be guaranteed later in the project. Where possible record where the input came from and the reasoning for it. Then if it turns out to be

Process	*Activity*
	impossible to meet, the risk of the situation can be evaluated.

Start up

As the project starts up it is generally necessary to revisit all the information from the Feasibility Study and hence start to better define the information that was recorded for it. The assumptions from the study need to be challenged and the accuracy of estimates refined. This is as true for Capacity as for any other area of the Project's make up.

Process	*Activity*
Estimate Volumes	The volume figures gathered from the business case are refined as the case itself is better understood. Start to challenge the assumptions, and start to add detail based either on business knowledge or "best guess" estimates. If information isn't available which is necessary to progress document an assumption and move forwards. This assumption should be revisited later on.
Investigate Technology	As likely ways forwards are progressed then try to gather performance related data. Make the data as specific as possible. Document assumptions where it isn't possible to generate specific data. It should be possible to get an idea of whether any given technology is likely to scale to your prospective volumes, and what it may take to do so.
Evaluate Risk	Further refine the risk analysis and take action on it to minimise the risk the project is carrying. Challenge the risk profile to make sure that it is representative, and where too much risk is being carried forwards decide what action needs to be taken to reduce it.
Performance Model	Start to work out how to build a performance model that will carry the project through into production. Choice of the model's complexity is important to the overall effort required in calibrating and maintaining the model, as well as in the accuracy of the results. As a rule, very complex models should only be considered where performance is truly critical and the cost of the hardware required to meet it is significant. Less accurate models may be used where it is cheaper to add extra hardware than to perform more detailed modelling. Calibration of a model will be difficult at this point. The easiest approach is to use a model to set a 'budget' for the overall performance of the system. Make sure, however, that any such budget is realistic or the result will be felt later in the project. Better to estimate a high hardware requirement now and pare it down later than the other way round.
Build and Test	Consider whether a form of performance test prototype is needed at this point. If the cost of the required hardware is high, or there is no real information about likely system performance then a

Process	*Activity*
	prototype is probably worthwhile. This has the added advantage of allowing the test and calibration processes to be prototyped as well. It is sometimes considered not worth prototyping unless the full production code and environment are available. I would generally recommend that testing is started early but that a degree of scaling effort is required, and the level of confidence in the model will be lower. This is still better, however, than no experimental results at all.
Manage Service Level	This is the best point to really get the business to consider the service levels that are appropriate to the final system. It must be clear that if the levels are set too high then the result will be significantly increased hardware and development spending. If the level is set too low the most likely result will be a project delay as they realise how exposed the business will be late in the project, and then raise the bar at a later date. If this doesn't happen the system will be deemed to have poor usability by customers.

Build

This is the most important part of the project – whilst the system is being built. If performance issues are found early on then they can usually be resolved with a fairly minor change of design and the result worked through the whole system. Unfortunately this is also true for all of the other factors in a project, such as creating the functional system. For this reason the performance testing that capacity planning is based on is often skipped. The result is usually a delayed production delivery.

Process	*Activity*
Estimate Volumes	In the fantasy project the volumes are fixed in stone well before this, and will never be changed. In all real projects the business are liable to be adjusting the business case, and hence the volumes it involves, through out the Build. As the implications of the system are better understood the basis for the volumes will be challenged and reworked as well. By this point you will be in a stronger position if each 'adjustment' can be argued to be a change – but even if it can't then manage it as if it is. This includes feature changes for the key features of the system.
Investigate Technology	Hopefully the high level technology in use by the system is mostly defined by now. There may still be areas that can be optimised, however, such as introducing a cache or system optimisation not previously considered. Again, manage these as internal changes even if they are not formally project changes.
Evaluate Risk	This should be considered as near continuously as possible. The changes of the system should be considered as to how they adjust the project's risk profile. If the risk profile changes significantly develop an appropriate action plan to deal with the new situation.

Process	*Activity*
Performance Model	As the system is constructed perform appropriate tests to calibrate the model to the actual system. If the real system exceeds it budgets then either start the process of optimising the system or change the budgets. For this approach to work at its best the system will need to be built vertically (function-by-function), but that is consistent with modern agile development approaches.
Build and Test	The key here is the "and Test" part. If the performance of the system needs a design change then the earlier this is found the better. The testing of an early version is likely to result in load test script rework, but this is generally less costly than project delays due to a non-performing system.
Manage Service Level	As for the Volumes, the required level of Service is also likely to be undergoing a level of churn. The expect level will general get more stringent with time, unless a strong manage keeps underlining the cost and time delay that the extra level will cause. If a model is available that can be updated quickly and efficiently as the different levels are considered then judgements will be better informed.

Formal Test

As the functionality is completed and tested the system begins to be tested functionally. It should also undergo a number of operational tests, one of which is a calibration sets of tests for the performance model. This is the basis of projecting how the system is likely to behave in the future. Lastly there should also be a set of capacity acceptance tests.

Process	*Activity*
Estimate Volumes	By now the expected volumes in the short and long terms are (hopefully) known. The projection should be sufficiently long term as to be able to judge the system's lifetime cost over a reasonable term.
Investigate Technology	The technology to be used is also (hopefully) known. Changes should only be considered where there is definite need, and then only under change control.
Evaluate Risk	The risk profile of the system should also be understood in the short and long terms. A reasonable action plan should have been agreed, and the profile and actions revisited where changes are agreed.
Performance Model	The performance model will focus the performance testing that is necessary. Once calibrated, the model should be used to predict the behaviour of the system under various loads and in different loading patterns. The model is then validated by testing the system in these situations to make sure that its predictions are accurate.

Process	*Activity*
Build and Test	There needs to be testing to calibrate the model, which can often happen in an environment that isn't live-like ("in the small"). There should also be load testing in a live-like environment ("in the large") that reproduces the situations envisaged by the modelling to confirm that the model is accurate. Other tests involve making sure that the system will operate consistently for an extended period ("soak testing") and at production data volumes ("volume testing").
Manage Service Level	Ultimately the target of formal testing is to demonstrate that the system will operate correctly for an extended period under predicted loading. This is sometimes not achievable, and so the next possibility is to demonstrate that the system will behave correctly in the short term, and estimate a time-span over which further action is needed to resolve future performance issues.

System Operation

Once the system is in production the Capacity Management starts to gather information from the actual system in operation. It is not unlikely that actual user behaviour will bear little resemblance to that predicted in the formal testing. Some agility may, therefore, be needed during the production operation. The management of the system's capacity becomes "business as usual" but must not stop. If the capacity is not monitored and managed then long-term performance problems will be the result.

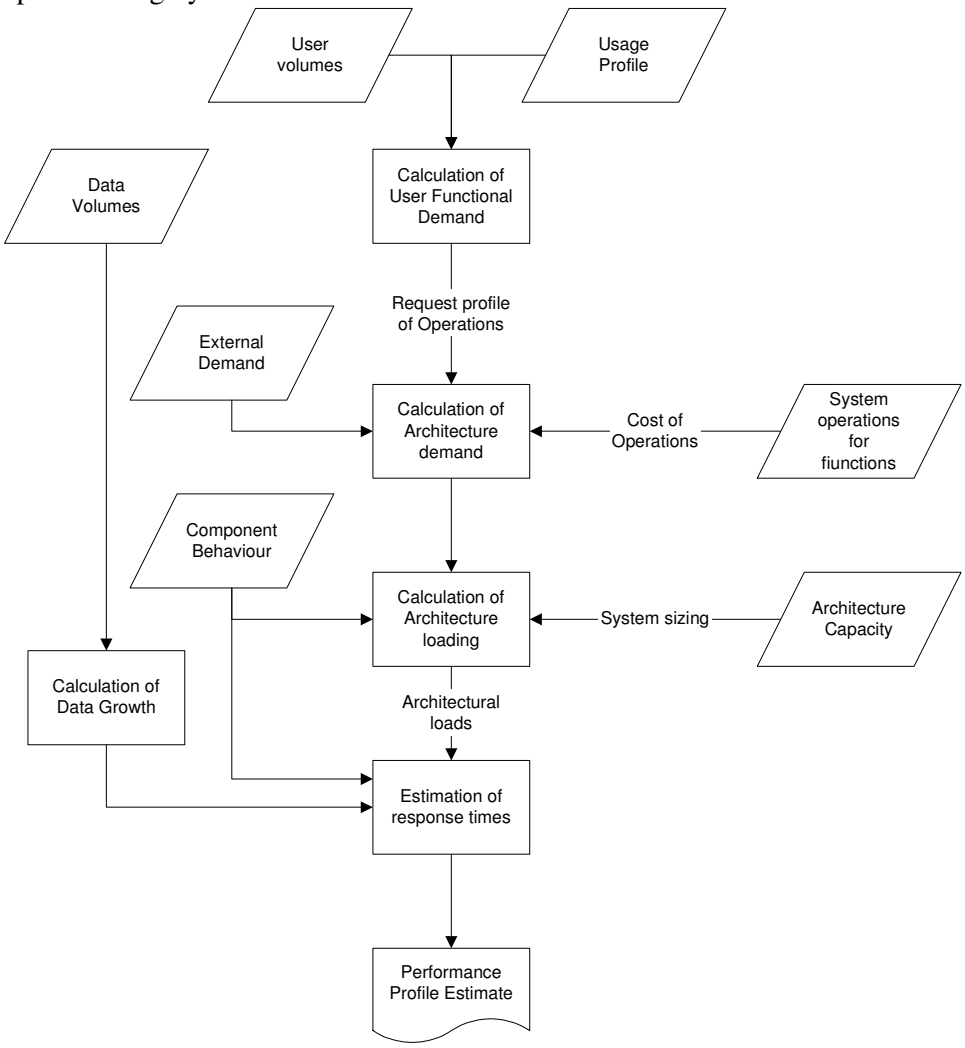
Process	*Activity*
Estimate Volumes	Compare the expected volumes over time with those being achieved. Make sure that the system's rated capacity is not expected to be exceeded. If it is likely then project the time window available to fix any issues in without causing the SLA to be compromised.
Investigate Technology	Maintain the system components and make sure that they are behaving as expected. Make sure that system patches that are applied do not compromise overall performance. Make sure that patches that are available and needed are applied.
Evaluate Risk	Make sure the risks to the system's operation are evaluated on a regular basis. Regular or continual risk management for the system should be "business as usual".
Performance Model	Make sure the model's expected behaviour reflects production experience. If not, investigate why there is a differential and resolve it by reworking the model. The re-project the likely system behaviour over time. This allows the risk that the system's SLA will be breached to be evaluated.
Build and Test	It is likely that continual system patching will require further testing, system optimisation and deployment. These cycles should also be used to maintain the view of the system's performance

Process	*Activity*
	into the future, and as an opportunity for optimisation where that is necessary.
Manage Service Level	The level of service being experienced by users needs to be understood and monitored. There are many mechanisms for doing this, but what is necessary is for the user experience to be compared with the experience that was agreed for them in the SLA. It is better if this is done automatically, rather than by user interviews. If the information is gathered by asking users then the chances are that performance issues won't be recognised until they are resulting in complaints.

3. Performance Modelling and Testing

Performance Model outline

The development and processing of a performance model would generate a model that operates roughly as shown below:



Performance Testing

The performance tests are intended to be able to do two things:

1. To generate the information needed to be able to calibrate the model. In order of priority the data to be generated is: Component behaviour, Systems operations for function, External Demand, and Usage profile data. Where data can not be generated experimentally the necessary values can be estimated from the information that is available.
2. To validate the model. Here the Performance Profile estimates generated by the model need to be checked against the performance profile that is being observed in a test or production system.

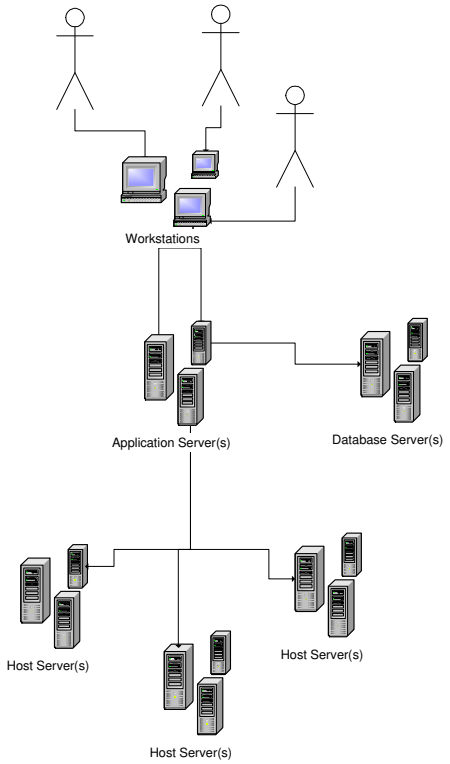
In the ideal situation, a set of dedicated laboratory-style data would be used for system calibration of component behaviour. The System operations for functions would be extracted based on a detailed knowledge of the system's design and development. The User Volumes, External Demand, and Usage Profile would then be extracted from a production system and projected into the future based on business knowledge. Lastly, data from test and production systems would be used to validate the performance model, and hence provide confidence in its predictions.

In the situation of this project this is not going to be feasible, and in reality rarely is. Thus the information that is currently available will have to be used, and a level of estimation added to this. As the project progresses more information may be able to be added into the model, and hence the confidence in it enhanced.

Ideal performance testing

Architecture

The following discussion assumes an N-tier client / server architecture, such as that illustrated below:



Ideal test laboratory

The following outlines the ideal test environment for performing a set of characterisation and model validation tests:

System	Description	Tool set
Interactive client machines	A specially introduced machine to allow manual client interaction. This would be used early on in the process as part of the system characterisation.	All client software. All activities in the client would result in data that could be thoroughly analysed at a

System	Description	Tool set
		later stage. Detailed operational performance statistics would be collected from the OS and the application for later analysis.
Load injector and management clients	Specially introduced machines that have a load generation capability to significantly more than the projected future level of simultaneously connected users.	Scripting and load generation tools, such as Mercury's load-runner tools, or OpenSTA in a Web environment. Detailed operational performance statistics would be collected from the OS and the application for later analysis.
Client network probe	A separate system configured and set up to capture all network traffic between the client machines and servers.	Detailed network capture and analysis tools.
WAN and probe	Client to server layer of networking	As for client network and probe.
Server network and probe	Server to server and server to host networking	As for client network and probe
Server and host systems	These would be set up with production-like capability. It would be practical to adjust the available data volumes and population distributions at will. It would be practical to add and remove CPU capacity, memory, disk space etc. at will.	Detailed operational performance statistics would be collected from the OS and the server applications for later analysis. All software would be fully monitored, so that database table locks used during particular client operations would be able to be monitored.

Other features of this ideal environment would include:

- All clocks would be precisely synchronised, so as to make analysis of output data simpler.
- The environment would be dedicated, with no external influences, such as shared use of the machines or networks. It would then be possible to add external load in a precisely controlled fashion.
- All data from all software would be able to be collated together and analysed in a single piece of software.

Such ideal environments are rarely, if ever, available. Thus, performance testing and management need to work with what is available in most cases. Pragmatism is a requirement, but does impact the time taken to generate results and the confidence that can be achieved in any conclusions.

Ideal UAT and production environment

Similar monitoring mechanisms to those described for the laboratory environment would be available. There would be a level of robot-based monitoring so that known operations were performed regularly, to allow a strict comparison of actual performance with a baseline value. All clients would be time-monitored against the SLA, for analysis purposes.

All operations exceeding the system SLA would be recorded. An alert threshold would be set so that any significant or regular pattern of SLA violation would result in an operational alert.

All data from the production and UAT environments would then be checked against the predictions of the performance model on a regular basis. This would allow an increasingly accurate model over time, and for variations from the model to be identified and investigated.

Ideal Tests

In the ideal performance testing situation the following types of performance testing would be performed:

Test Type	Description	Result
Characterisation	Early in the overall test process a set of tests would be run at low volume. All monitored data would then be gathered and analysed in detail.	Structure of performance model, with initial calibration
Calibration	A comprehensive set of tests would be run on a production ready system. These would investigate in detail: Baseline resource usage, including average and statistical usage variation. Baseline testing under varying levels of system capacity and layout. Volume tests to generate information on resource usage variation with data volume.	Performance model calibration
Load Testing	Load tests to introduce controlled level of users with known usage patterns.	Performance model validation
Soak Tests	Running the system for an extended period (multiple days) under load.	Validation that the system's performance does not degrade in use.
Destruction tests	Running the system under loads far exceeding those expected in production.	Understanding of how the system will degrade. Validation that there are no blocks to overall scalability, or instability under loads above those of production.

4. Sarquol Limited

We work with companies with large IT infrastructures that are having performance difficulties with their systems. We then work with them to improve the performance of their systems using a combination of testing and modelling. This will result in more efficient business processes and more of the business user's time available to service customers rather than their own systems.

This document outlines the principles of the Capacity Management approach that we recommend to clients, and is used to help introduce clients to the approach. Please distribute it freely in its complete form.

If you would like access to other resources that we use as they are published then please register at our web site: http://www.sarquol.com. This will provide you with access to the site and the resources on it as they become available. As a member of the site we will also send you a bulletin on an occasional basis that is intended to support you capacity management efforts. There is also a forum available on the site where help can be obtained where time allows.

If you are involved with an organisation that has capacity management issues that needs more direct support then please contact us directly. The best start point would be to e-mail David Howard, at dh@sarquol.com, who will be happy to discuss the most appropriate way forward. If the requirement is urgent then please feel free to call him on +44 (0) 7887 536083.

End of Document