Setting up and Managing a Test Lab

Author Ruth Keys, Independent Testing Consultant

Am Weissen Berg 1H
61389 Schmitten
Germany

http://www.keys-consult.de/
ruth@keys-consult.de

Abstract How you set up and manage your test environment for high level testing can contribute considerably to the efficiency and success of your testing efforts. Regardless of whether you are testing from your workspace or in a dedicated test lab, planning and organization are needed before you can kick off your test execution. Not only do you need to think about the physical equipment required, but also organizational issues must be addressed: how to prevent cross-contamination of testing efforts, how to move efficiently from one release to the next, how to organize test data.

What is a test lab?

Before launching straight into the business of setting up and managing a test lab, let's define what a test lab is. Two definitions taken from Rex Black's book "Managing the Test Process" [1999] will help us here:

> "A test laboratory is a place where testing is conducted."

This may seem to be stating the obvious, but in fact encourages us to consider every setting in which testing takes place as a test lab. So even if you are testing from your normal workplace, planning and organization are needed before you can begin your test execution.

> "Testers must perform controlled experiments, dealing with measurements and known quantities."

This second statement appeals to my scientific background. A test lab must provide an environment in which to conduct these controlled experiments. And the environment must stable, ordered and reproducible.

Having been responsible for establishing test labs in 2 large projects, I will draw on my experience to provide practical suggestions and some tips and tricks from the real world, which should jump start test managers new to this area and perhaps provide some new ideas to old hands. Apart from the theory, the paper also covers a case study from one of the projects.

Five Ws and an H

Setting up a test lab from scratch can be a large and complex task. It is in fact a small project and as for all projects it helps to have a plan. The five Ws are the five planning questions: Who? What? When? Where? and Why? For this project we need to add an H: How? Answering these questions will help us to state our goals and then define the ways and means to achieve them.

It is important to be clear about your objectives. The obvious goal is to set up a test lab. But to be more precise answer these questions.

WHO?

- Who will be doing the testing?
- Who will be using the test lab?
- Is the lab for one project? The whole organization?
- Will more than one test team need to be accommodated?
- How many people at any one time?

WHAT?

- What will they be testing?
- What type of testing will be performed?
- Functional acceptance testing?
- Performance testing?
- User acceptance testing?
- Operations acceptance testing?

WHEN?

- When do you need a fully functional test lab?

- Is a staged implementation possible?
- Are there any other time constraints? e.g. delivery dates for hardware

WHY?

- Why do you need a test lab?
  Be prepared to answer this question! Setting up a dedicated test lab isn't cheap. You will need to justify the expense to your managers.

To decide whether a virtual or a physical test lab is required there are again questions to help:

- Can all the testing be performed in a normal office environment?
- Are there particular hardware requirements?
- Are there particular software requirements that are not part of the standard workplace or which cannot be installed there?
- Are there special security issues?
- Does access to the test environment need to be restricted?

Armed with the answers to all these questions you will be able to concentrate on answering the questions What? and How? What resources are needed to set up the test lab? How should are these resources managed? – What processes are required? The rest of this paper concentrates on the *what* and the *how* of setting up a dedicated test lab. Even if it has been decided that testing can be conducted from the normal workplace the majority of the following suggestions need to be considered and implemented too.

The test lab inventory

The test lab inventory covers all the facilities and physical resources you will need in your test lab.

Space

The most important resource in the test lab is space. It is imperative to resist the temptation to stuff as many test workspaces as possible into the lab. Don't forget that the testers will be working in the lab 8+ hours a day! They will not thank you if they are sitting in a broom cupboard with no space around them to spread out papers and pencils. Testing is not just about using the software, testers also need to refer to test plans, test scripts, take notes etc.

Office furniture

Just as the testers are unlikely to be motivated if confined to the least attractive office in the building, they will also not appreciate the discarded office furniture. Have good quality office chairs and tables. There should also be plenty of cupboard space. It is good practice to keep all physical test results (hardcopies, used test scripts, floppies etc.) in the test lab. Make sure that there are an adequate number of telephones in the test lab. One phone is rarely enough!

Other invaluable aids in the lab are flipcharts, a pinboard, and a whiteboard – useful for having pieces of vital information visible for all to see. A good example of this is information which is likely to change e.g. if you change the system date from the actual calendar date – write the current date in the test environment in large letters on the whiteboard.

Lighting / Air conditioning

If you will have a large amount of hardware producing waste heat then air conditioning is a must. Not only do you need to keep the hardware at a controlled temperature, the testers will thank you if they aren't subjected to an involuntary sauna every time they have to work in the test lab. It may be pleasant in the winter, but come summer and no one will be rushing to test.

Similarly do consider the lighting carefully. I have had to work in a test lab in the basement with no windows. It is not an experience I particularly wish to repeat. Whereas no windows is the one extreme, too many can be the other. Especially if they are without blinds to reduce reflections on the screens. A similar problem can also occur with overhead lighting.

(Computer) hardware

Obviously you are going to need PCs or workstations. Are there any other particular hardware requirements? Do you need special printers? Are you testing a Computer Telephone Integration (CTI) application with special telephony requirements? Are other special peripherals required?

Once you have your shopping list for hardware, be sure to check that the room proposed as the test lab has enough electrical power sockets and network connection points and that they are in the right place. Having to run extension cables around the room is an accident waiting to happen.

Office supplies / stationery

Ensure that the test lab is stocked with adequate supplies of:

- Paper (notebooks and printer paper)
- Pens, pencils, sharpeners, erasers, stapler, hole punch
- Post-its, sticky tape
- Printer toner, floppies

Don't forget that your testers may only be on assignment for a short time and not have access to the normal channels for office supplies. Keeping a supply on hand in the test lab prevents frustration and helps keep your testing on schedule.

Reference material

This includes:

- Test plans, test scripts, test case descriptions
- User manuals for hardware and software, including the manual for the application under test, if available
- Telephone list – with the phone numbers of all the important contact people in case of problems
- Checklists, pointers to the nearest photocopier, coffee machine etc.

Door lock

If you have expensive equipment, sensitive data or anything else you don't want disappearing from the test lab – be it only the printer toner – then it is a good idea to have a lock on the door. You may also not only want to stop stuff leaving the lab but to prevent people getting in. On one project the test lab PCs were very popular with the developers for running their long build process.

Establishing a controlled environment

Once the hardware is set up, the next step is to define the software requirements.

Baseline installation

For high level testing, the PC or workstation should reflect as closely as possible the production environment, where the application will be used once deployed. Not only should the operating system and middleware be considered, it is also important to have any other applications installed which will be used and exist concurrently with the application under test (AUT) in the final live environment. At the very latest in the User Acceptance Test the application must be tested running in parallel to other applications to rule out any compatibility issues.

Test tools

Decide which test tools will be installed where. Although some test tools will have to be installed on the test PCs e.g. capture/replay tools, one should be aware that every test tool that is installed takes you one step further away from the live environment. It is worth considering having a separate PC in the lab that provides access to test tools not directly related to test execution, e.g. test management tools, bug tracking tools.

Application under test

It is unlikely that you will be able to get away with only one installation of the AUT, unless you have a well-established system that is in the maintenance phase of its life cycle. So you will need to decide how many installations are required. I would strongly recommend that performance testing be separated from functional testing. It is worth considering having separate installations for different testing groups, i.e. functional acceptance testers, user acceptance testers and operations acceptance testers should all have their own installations.

Other constraints to consider are the number of releases it may be necessary to test in parallel and whether the installation can easily be made to access different databases. The latter, if possible, may enable you to reduce the number of installations by assigning separate databases to the different test groups and using time slots for the testing.

Other requirements

Will development be given direct access to the test environment? There is undoubtedly a need to be able to transfer information, such as trace files, log files, protocols, from the test environment to the development environment in electronic form. There are a number of possible ways of achieving this, including e-mail, a LAN connection, CDs, memory sticks.

Finally you are going to need test user accounts. These allow you to assign different roles and responsibilities to the user. Without this flexibility you will be unable to properly test all the functionality of the application.

Keeping chaos at bay

Having defined the requirements for the environment and probably done your first installation, now is the time to start getting your processes defined. In the heat of the testing (who-ever heard of calm, leisurely testing?) if you don't have well defined and practical processes, you will soon find your test environment deteriorating into an uncontrolled chaos.

These are some examples of activities you may want to have processes defined for:

- Installing a new release of the AUT
- Upgrading the baseline installation
- Applying patches to the installation
- Test data load and maintenance
- Taking backups of the testware
- Booking time and resources in the test lab

The processes should be as practical as possible. Try to eliminate all unnecessary bureaucracy. It also helps to have the processes documented in a form, which is easy to refer to and understand at a glance. I have found that RASCI charts are a good way to do this. A RASCI (responsible, approves, supports, is consulted and is informed) chart is a simple tool that can be used to identify roles and responsibilities in a process to make sure ownership, approval, support, consultation and communication responsibilities are assigned. See Reid [2003]

It is a good idea to collect all this information together into a Test Handbook, which of course then belongs to the reference material that is made available in the test lab. I have found it extremely useful to make checklists (excel sheets), which list the activities, in the order they are to be performed and have tick boxes for each task to record progress (very useful if you get interrupted by something e.g. a phone call).

The checklists belong to my philosophy of making important information readily available to everyone. If you run out of time when documenting your processes then just make the checklists. It is a simple truth that anything, which has a practical use and adds value, will be kept up to date.

Similarly I make good use of whiteboards in the test lab to display pieces of transient but vital information. Post-its are put to use to mark up the test PCs with information such as which operating system is installed, which installation of the AUT they are hooked up to, etc.

Better data, better testing

I quote James Lyndsay in heading. This was the title he gave to a tutorial he held at EuroSTAR 2002. And I can only agree with him wholeheartedly. If you think you've finished, when you have defined your requirements and processes, then think again. You need to put some serious thought into managing your test data. Controlling test data is part of dealing with known quantities, which in turn leads to reliable test results.

Test data defined

Test data can be divided into 3 types, Lyndsay [2001]:

- Environmental data
  that tells the system about its technical environment, e.g. directory paths
- Setup data
  that tells the system about the business rules, e.g. tariff structures
- Input data
  is entered into the system by normal functions of the system.

I like to subdivide input data into 2 categories:

- Primary input data
  that is entered into the system during test execution as part of the test case.
- Secondary input data
  that exists in the system before a test is executed and that affects or is affected by and is a prerequisite for the test execution.

Environmental data is loaded manually, usually during installation. It often causes lots of problems the first time you try to install the AUT in your test environment and keeps you in constant touch with development, while you try to get a clean, functioning installation. Be sure to keep good notes during this painful birth and document it all. Setup data is also usually loaded manually using a system administration tool associated with the application. Input data is naturally part of test case design.

Environmental data and setup data are normally fairly stable and not subject to frequent change. Input data is transient data and therefore more likely to decay over time and during test execution. Although all types of test data need to be managed, it is the input data that will require most attention.

Managing test data

Even if you have multiple installations you will still have more than one tester using the test data in an installation. To prevent cross-contamination of test results caused by multiple updates to test data, it is sensible to divide the test data into logical data sets and to assign a particular data set to a tester. The data sets must be defined so that data used in one area, will not affect the results of tests in another area. Typically the data sets are defined at the level of a business object. Customer data may be divided into logical sets based on a range of values for the customer number. Product data can be divided based on product groups, e.g. one logical set contains all kitchen appliances, another all bedroom furniture.

Another useful concept is that of test data partitions, Lyndsay [2001]:

- Safe area
  No test changes the test data, so the area can be trusted.
- Change area
  Tests may change the test data, but must reset or reload the data after testing.
- Scratch area
  Existing data cannot be trusted, testers use at their own risk.

The use of data sets and partitions rely on the discipline of the testers in sticking to the rules. They are not in themselves a foolproof way of preventing data corruption.

When defining the processes for test data management it is important to decide where the test data will be stored. Will it be inside or outside the test environment? Having a well-defined directory structure for storing test data, which is intuitive to use, will make the data readily available to everyone and also help prevent data contamination.

Another consideration is the format in which the data is stored. There are a number of possibilities here, e.g.:

- In a neutral format (excel sheets)
- Input format/files for utilities
- Using import/export functions of the database.

A case study

In a project for a service provider to the travel industry I was involved in system testing a client/server application for travel agencies. The system test covered functional acceptance testing, usability testing and performance testing. I was managing the test team performing the functional acceptance test. The test lab that we used for the system test ended up being used by the whole organization, for all client/server applications, for system testing, user acceptance testing and operations acceptance testing.

First steps

We started out with a virtual test lab – in other words we were testing from our normal workplaces. We were also using a workaround provided by the developers which allowed us to test without using the middleware, which was proprietary and, when the system test started, non functioning. At this stage we were five testers doing the functional acceptance test and one tester concerned with usability aspects. We had our own installation of the AUT (minus middleware) and database.

We also had our first experience of tests corrupting the test data for other testers, so we quickly learned that we needed logical data sets to isolate the tests of one tester from another. The logical data sets were defined by value range per business object. Our test data was rather primitive, as we needed secondary input test data in the system, for which there were no function as yet developed for data entry. The developers had written themselves a simple utility to load this type of data, which we also used in our test environment. Some test cases produced global data changes. Having exclusive time slots for the execution of these tests solved this problem.

Test lab established

Clearly there were a number of problems that needed resolution before much more testing was done. It became our job to communicate with the developers of the middleware (another project in the organization) to establish a test environment that was much nearer to the production environment. One member of the team took on the responsibility for this. At the same time it was decided that we needed a dedicated test lab, as there were hardware and software requirements that dictated this.

The test lab was configured as follows:

- 6 PCs with interchangeable hard disks, 2 per PC with different versions of the Windows operating system
- 2 gateways (X25 and X31)
- 1 PC as a repository for the testware with a DAT drive for taking back ups
- 1 PC monitoring the server
- in an isolated section of the LAN.

We also had the middleware in place and were joined by 2 more performance testers. So we needed a second AUT installation as otherwise the performance testers were confined to the early morning shift and the functional testers to the late shift. The second installation was not all that long in coming, although originally we managed with 2 databases, one installation and time slots.

The quality of the test data was improved with the delivery of a migration program. In the course of testing this new functionality we were able to load better quality test data into the system. The test data was kept in neutral format as input data, which could be converted, using a test tool, into the input files necessary for the migration program. The test data was easy to modify and it was simple to generate larger volumes of secondary input data.

Processes in place

The beginning of the user acceptance tests and the operations acceptance tests heralded the next improvements. The operations acceptance test (OAT) team also began testing from their workplace, but soon decided that they too needed a test lab. From them we learnt about Ghost images and they helped us to improve our processes for performing baseline installations using Ghost imaging software. We were able to restore the baseline, which included the operating system, MS Office application and our test tools, in a fraction of the time needed previously.

The OAT team also helped us with our test data management. They had people who were knowledgeable in the use of the database utilities and we benefited from this by using their scripts for saving and loading test data sets from and into the database.

The user acceptance test team shared our test lab. Although not professional testers, they were open to using test tools and needed very little persuasion to use our test management tool to document their test cases and test execution results. We separated their tests from ours by having two databases within one AUT installation.

When the service provider moved buildings, it was decided that there should be one larger test lab, which would be administered by the OAT team, who were now the operations department for client applications within the entire organization. All future projects using client/server architecture would be required to do their high level testing in this test lab.

Lessons Learned

From the project just described and from further projects where I have been involved in establishing and managing test labs I have learnt that to avoid being overwhelmed by the size and complexity of the task at hand it is best to start small and evolve with time and experience. This approach allows you to avoid planning paralysis – where you spend so much time planning and defining that you forget to get started.

A useful model to help you with the continuous improvement of the running of the test lab is the PDCA cycle originally conceived by Walter Shewhart in 1930s, and later adopted by W. Edwards Deming. The PDCA cycle consists of four steps Plan, Do, Check, Act. Each turn of the cycle leads into the next. The PDCA cycle is a good method to assist you in growing from a small to a large outfit.

Be creative about tapping outside resources – you don't have to do everything yourself! Talk to

- the developers. Do they have tools, scripts, data you can use?
- database administrators. Get them to help with automating import/export of test data, restore/recovery of testware.
- PC Support groups. They often have automated routines to set up baseline PC installations.
- Operations. Will they administer the test lab too? Or at least offer advice.
- OAT team. Will they help with the administration of the AUT? – as a chance to get familiar with the software before they start their testing.

I chose the case study as I felt that particular project demonstrated most of these points.

Deal with the difficult,
While it is still easy.
Solve large problems
When they are still small.
Preventing large problems
By taking small steps
Is easier than solving them.
By small actions
Great things are accomplished.
Lao Tzu

References and Further Reading

Black, Rex. 1999. Managing the Test Process, Microsoft Press

Dustin, Elfreide, Jeff Rashka, John Paul. 1999. Automated Software Testing, Addison-Wesley

Fewster, Mark, Dorothy Graham. 1999. Software Test Automation, Addison-Wesley

Lyndsay, James. 2001. The Importance of Data in Functional Testing. Available at http://www.workroom-productions.com/

Reid, R Dan. 2003. ISO 9000 and more, Quality Progress, Milwaukee, Apr. 2003

Reizer, Neal and Nancy Wang. 2000. Juggling Concurrent Releases, STQE Volume 2, Issue 6 Nov/Dec

Links to Resources

FinePrint – prints A5 booklets
http://www.fineprint.com

Ghost Imaging Software
http://www.symantec.com/sabu/ghost/

Hummingbird - Network Connectivity Solutions
http://www.hummingbird.com/products/nc/index.html

StickyMinds.com Message Boards
http://www.stickyminds.com

TOAD® - an intuitive graphical user interface to Oracle
http://www.quest.com/toad/