APPLICATION DELIVERY

MERCURY LOADRUNNER BEST PRACTICES GUIDE FOR R/3

MERCURY™

INTRODUCTION

The Mercury LoadRunner® Best Practices Guide describes Mercury's recommended LoadRunner implementation methodology and typical issues encountered in an R/3 implementation.

An R/3 system incorporates a number of moving pieces – enterprise network, client machines, application servers, Web servers, database servers, etc. Most if not all large-scale SAP implementations suffer from performance-related problems. These problems range from poor online transaction process response times to inadequate batch throughput. Problems may arise for any number of implementation issues, such as inaccurate volume/sizing estimates, undefined operational profiles, poorly written client development code, un-tuned database, hardware configuration, or an un-tuned operating system.

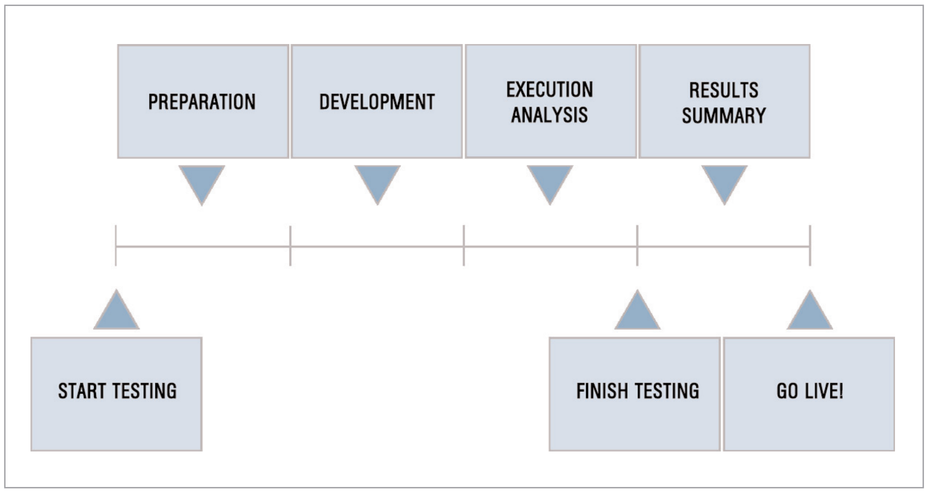
The goal of a comprehensive load test is to proactively diagnose and eliminate these performance problems. This is accomplished by running model loads in a test environment, identifying system bottlenecks, and addressing them before a system is brought into production.

TABLE OF CONTENTS

Best Practices Process

The typical best practices process for LoadRunner includes four phases: preparation, development, execution, and results summary.

- **Preparation.** Prior to using LoadRunner, the project manager performs such preparation tasks as planning, designing, configuring the environment setup, and completing product training.
- **Development.** The second phase involves creating the scenario and test scripts that will be used to load the system.
- **Execution/Analysis.** The third phase includes running the scenario. The data gathered during the run is then used to analyze system performance, develop suggestions for system improvement, and implement those improvements. The scenario may be iteratively rerun to achieve load test goals.
- **Results Summary.** The purpose of last phase is to report the outcome of the work performed for the load test.



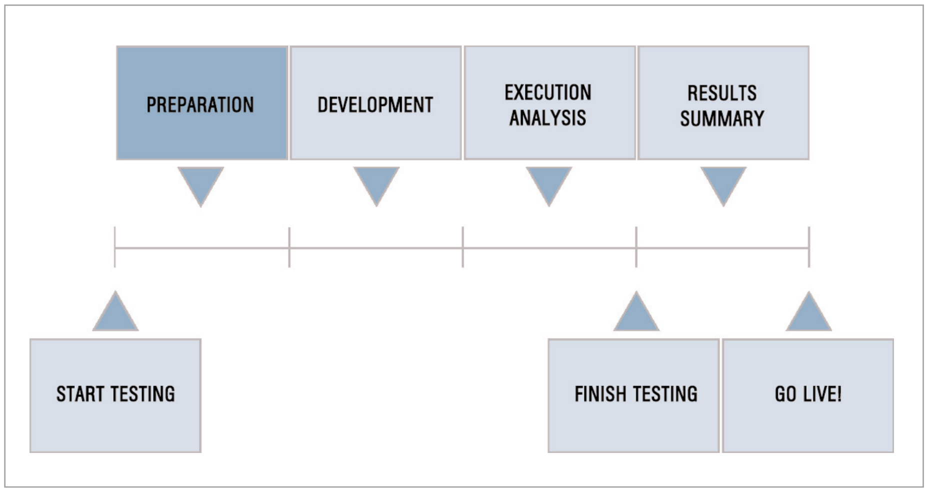
An example of the LoadRunner best practices process.

The complete best practices process involves accurately modeling system load, developing tests to create that load, driving that load against the system, debugging system issues that are identified, and iteratively rerunning tests. This process provides a means to fully test the system and develop considerable confidence in system performance. It typically requires a minimum of three months to complete.

Similarly, the abbreviated best practices process is used to perform a sanity check of the system. It quickly gathers preliminary results about system performance and reliability with limited planning and a small number of representative scripts. It typically requires four weeks to complete.

This next section will describe the best practices process in more details. Later in this paper, the abbreviated best practices process will also be described.

Preparation



Preparation is the first step of the best practices process.

The first step in a successful implementation is to perform preparation tasks before using LoadRunner. It includes planning, analysis/design, defining "white box" measurement, configuring the environment setup, completing product training, and making any needed customization. The sample LoadRunner implementation provides a boilerplate to assist in this effort. The preparation phase can typically be completed in one month.

Planning

The first step for testing is to define the implementation goals, objectives, and project timeline. The output of this process should include writing a planning document that defines all issues going forward. The sample LoadRunner implementation contains an example.

The first element of the planning document should include the project goals. The goals broadly define the problems that will be addressed and the desired outcome for testing. A typical goal would be:

"Company ABC is considered to be a high-risk SAP implementation based on its high transaction, printing, and data volumes. The goal is to conduct a series of load tests in order to identify performance/operational bottlenecks and recommend improvement measures."

The second element is to outline the project objectives. The objectives are measurable tasks that, once completed, will help meet the goals. A typical set of objectives would be:

- Estimate the concurrent user utilization and transactional volumes for the FI, MM, and SD modules for the North American business unit.
- Emulate that utilization and measure system response time.
- Identify performance/operational bottlenecks and recommend performance-improvement measures.

Next, a project outline will need to be developed so that the sequence, duration, and staff responsibility for each implementation task is clearly defined.

In summary, project managers and/or technical leads typically perform the planning phase in conjunction with the implementation teams.

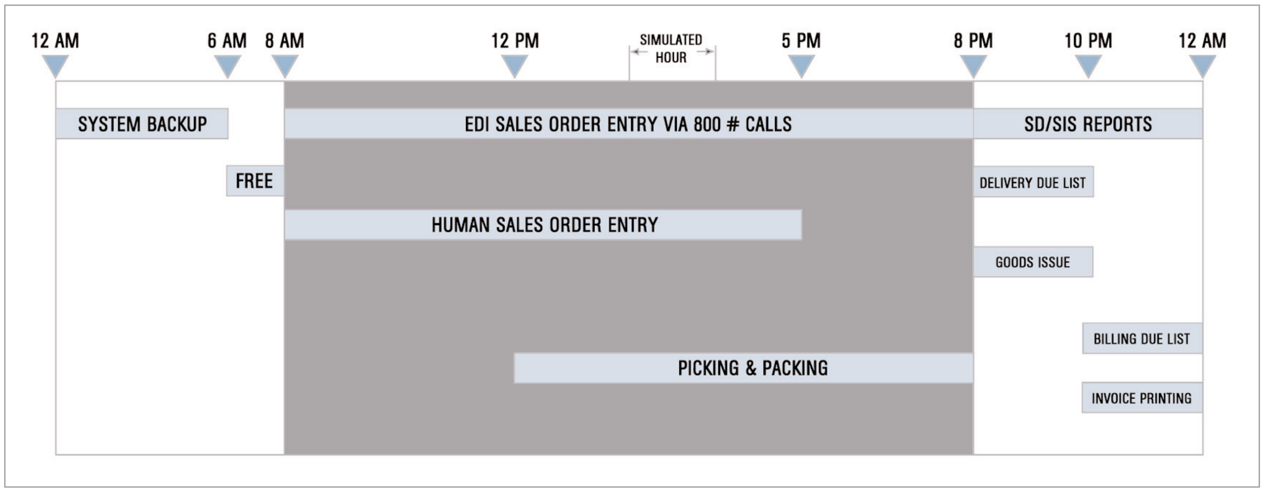
Analysis/Design

The analysis/design phase specifies the tests that will be constructed. The design should include scenario definition, user description/activity, transaction description/volume, step-by-step transaction semantics, and transaction data.

In this context, the analysis/design should first identify a set of scenarios that model periods of critical system activity. Examining the operational profile of the system derives this. This analysis is especially important in global operations where one continent's batch processing is running concurrently with another continent's online processing.

For example, in the following figure, one scenario should include testing system performance from 2 p.m. to 3 p.m. when EDI sales, Human Sales Order Entry, and Picking/Packing are all running concurrently. A second scenario might be from 8 p.m. to 9 p.m. when batch processing commences.

Typically each scenario will address one of the load test objectives.



A sales and distribution operational profile.

Shaded area is Day Time Online Processing.
Unshaded area is Night Time Batch Processing.

Each scenario will have a number of components. The first is the number and type of users:

	ORDER-ENTRY CLERK	SHIPPING CLERK	GL ACCOUNTANT	...
# Users	100	50	10	...

The number and type of users.

The second component includes the type and frequency of business processes for each user:

BUSINESS PROCESS	ORDER-ENTRY CLERK	SHIPPING CLERK	GL ACCOUNTANT	...
Create Order	10/hr			
Display Order	5/hr			
Ship Order		15/hr		
Generate Invoice			5/hr	

The business process transaction frequency.

The third component is to set up each business process in a step-by-step fashion. This includes notes for data input and performance measurement. These steps should contain enough information that an individual with limited application experience can work effectively:

STEP	DATA	RESULT
Enter the Create Order Screen Number in the OK Code	va01	"Create Sales Order: Initial Screen" should appear
Fill in the Required Fields	Order Type = <OT> Sales Org = <SO> Dist. Channel = <DC>	
Start Timing	"Single Line Entry"	
Hit Return		"Create Standard Order: Overview – Single Line Entry" should appear
End Timing	"Single Line Entry"	
...	...	...

A step-by-step business-process definition.

The last component is to define the data used by each business process:

<OT>	<SO>	<DC>
OR	1000	10
OR	3000	10
TA	0010	02
...	...	...

An explanation of the business process data.

The more detailed and accurate this information, the more reliable the stress test. Experience shows that it is effective to model 80 percent of the user population and/or transaction. It is difficult and time-consuming to achieve greater accuracy. This is typically modeled by five to 10 business processes per application module.

White Box Measurement

The white box measurement section defines the tools and metrics used to measure internal system-under-test (SUT) performance. This information helps to pinpoint the cause of external performance issues. It also leads to recommendations for resolving those issues. The following table describes an abbreviated list of common metrics for SAP R/3 with an Oracle database running on an HP server:

		CPU MANAGEMENT		
PARAMETER	**IDEAL REQUIREMENT**	**MEASURING TOOL**	**MEASURED VALUE**	**REMARKS**
Utilization	None of the CPUs should have utilization of more than 95 percent.	GlancePlus - CPU Screen	50 percent	High CPU idle time indicates a machine with resources to spare, warranting a downsizing.
Load Average	No more than three processes should wait for the CPU.	GlancePlus - CPU Screen	OK	
CPU Intensive Processes	Any non-Oracle, HP system processes utilizing CPU intensively warrant attention.	GlancePlus - Main Screen	OK	
		MEMORY MANAGEMENT		
PARAMETER	**IDEAL REQUIREMENT**	**MEASURING TOOL**	**MEASURED VALUE**	**REMARKS**
Swapping	Minimal to no swapped-out processes	GlancePlus - Memory Screen vmstat -S 5 9	None	Since the total system memory is limited, size of SGA and UNIX buffers would have a direct impact on swapping and paging.
Paging	Minimize page faults	GlancePlus - Memory Screen sar -p 10 10	Minimal to None	
Data Buffer Hit Ratio	More than 70 percent	CCMS	> 80 percent	Tune by increasing the DB_BLOCK_BUFFERS.
		DISK MANAGEMENT		
PARAMETER	**IDEAL REQUIREMENT**	**MEASURING TOOL**	**MEASURED VALUE**	**REMARKS**
High Disk Activity	Disk should have less than 50-70 I/Os per second. Also, the disks should be less than 60 percent busy.	GlancePlus - Disk Screen sar -d 5 2	Minimal	
Long Disk Queues	Minimal to no disk request queue. A queue of 2-4 warrants attention.	GlancePlus - Disk Queues sar -d 5 2	Minimal	
Balanced Disk I/O	Minimal spread in the disk activity among the Oracle data disks.	GlancePlus - Disk Screen Server Manager	Tablespace layout exercise required	
		DATABASE MANAGEMENT		
PARAMETER	**IDEAL REQUIREMENT**	**MEASURING TOOL**	**MEASURED VALUE**	**REMARKS**
Full Table Scans	No full table scans for long tables (more than 5 data blocks), if less than 20 percent of the rows are to be retrieved.	CCMS	About 5 percent of the total table scans.	Either creating or reordering the indexes on the table can avoid full table scans on long tables.
Unindexed Tables	All tables should have at least a primary index.	CCMS	None	

An example of the white box measurement metrics.

Environment Setup

The purpose of the environment setup phase is to install and configure the system under test. Bringing up the system is a requirement for any testing activities. Preparation includes setting up hardware, software, data, LoadRunner, and white-box tools. The system's business processes should function correctly.

Product Training

The product-training phase educates the personnel who will be using LoadRunner. This training includes classroom material and mentoring on the system. After the product training, the test team should then have the skills required to start implementing the design.

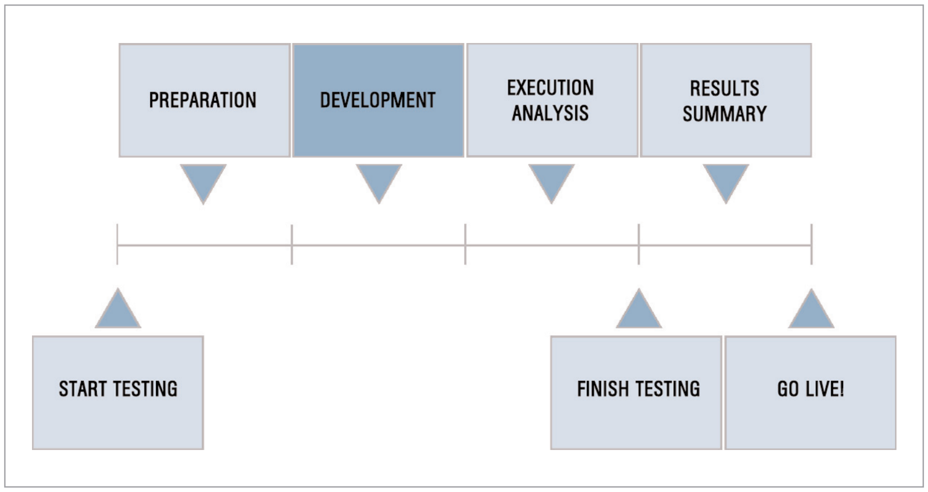
Optional Customization

Customization, the final phase of the preparation section, is optional. This is required for environments that LoadRunner does not support "out-of-the-box." For example, customization would include building a record/replay solution for a unique environment.

Development

During the development phase, the test team builds the tests specified in the design phase. Typically development requires considerable effort, typically one month of project time. This depends on the experience of the staff, the number of tests, test complexity, and the quality of the test design.

This estimate is based upon common "rules of thumb" and typical project size. New LoadRunner users who have just completed training can build one script per day, and experienced users can build two scripts per day. Typical projects have two to three individuals and 50 scripts. This also builds in time lost for such unexpected events as environment setup issues, incomplete design, etc. This is discussed in more detail under common implementation issues (See page 14).



The development phase in the best practices process.

Initial Script Development

It is desirable to have a high degree of transparency between virtual users and real human users. In other words, the virtual users should perform exactly the same tasks as the human users.

At the most basic level, LoadRunner offers script capture by recording test scripts as the users navigate the application. This recording simplifies test development by translating user activities into test code. Scripts can be replayed to perform exactly the same actions on the system. These scripts are specified in the design and should be self-explanatory. In any case, there should be functional support available to assist if there are any system functional anomalies.

As tests are developed, each test should first be built with one user, static data, on a local script development machine — the simplest case of script development. The script should be debugged to consistently replay. Dependencies (data consumption, correlation, preparation, cleanup, etc.) should be well understood.

Parameterization

Script recording alone is only a beginning. Replaying the same users' actions is not a load test. This is especially true for large multi-user systems where all the users perform different actions. Virtual user development should create a more sophisticated emulation — users should iteratively perform each business process with varying data.

Script development next extends the tests to run reliably with parameterized data. This process reflects the randomness of the user-population activity. Once virtual users run with development data sets, that data should be saved with the tests.

Multi-User Debug

Finally, tests should be run with a handful of concurrent users. This helps to resolve multi-user issues like locking and data consumption. Performing multi-user functional verification during script development will save considerable time during the execution phase.

Finally, scripts (with data, GUI maps, and any other needed component) should be saved into a "golden" script area. This may be a configuration management system, or simply a common network drive. The tests saved here will be used in the execution phase.

Build Volume Data

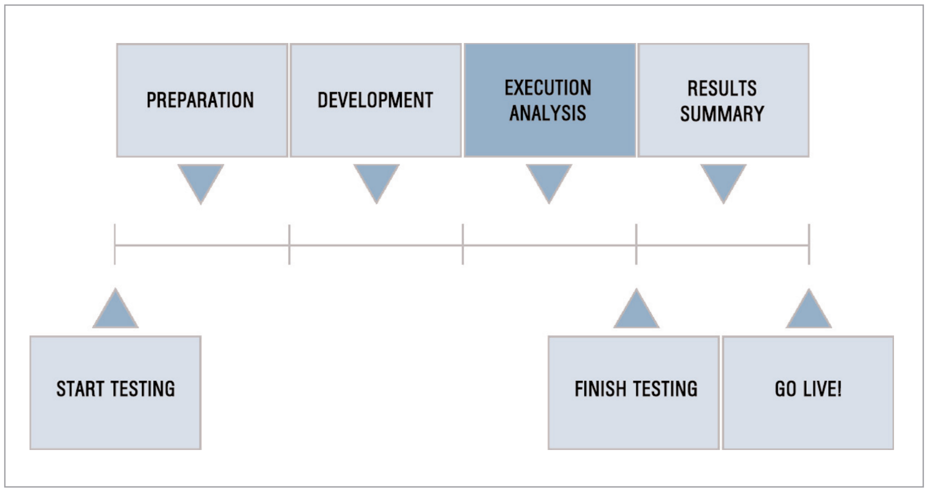
In parallel to script development, volume data should be constructed to support the execution of the load test. Typically business processes consume data — each data value may be used only once. As a result, there needs to be sufficient data to support large numbers of users running for a number of iterations — often 10,000 items or more.

Execution/Analysis

The execution/analysis phase is an iterative process that runs scenarios, analyzes results, and debugs system issues.

Test runs are performed on a system that is representative of the production R/3 environment. LoadRunner is installed on driver hardware that will create traffic against the system under test. This work may be done at the customer site or a hardware vendor benchmark facility.

Each scenario should run for at least an hour. This is done to gather statistically significant data on system performance. Any white-box tools used for internal system performance measurement should capture data during the run.



The execution/analysis stage in the best practices process.

The system should be loaded with enough data to support three or four scenario runs without a refresh. This should be done at the end of each day to give the team optimal productivity.

High-skilled individuals then monitor the system during execution. They use LoadRunner information and white-box information to analyze the run. Based upon this information, recommendations may be suggested and implemented on the system. This process is performed iteratively until the system performs up to expectations. This activity usually requires a dozen iterations to complete system tuning.

Typically the execution/analysis phase requires one month and should be completed one month before production system deployment. Execution/analysis is best separated into a number of phases to most effectively debug system issues that are detected.

Light Load

The first step is to install the system under test and successfully run the scenario's test scripts with a small numbers of users.

Since the scripts functioned properly in the development environment, the emphasis should be to recreate this functional environment for execution. This is important if "production-like" hardware is added to an in-house environment or if the project transitions to a hardware vendor benchmark facility.

Any "new" script execution errors will typically indicate system configuration differences. It is advisable to avoid script modifications at this stage and concentrate on system-under-test installation.

Moderate Load

The second step is to run a moderate load. This will help to flush out gross system issues that do not affect functionality. The moderate load should be 25 to 50 percent of the total user count or transaction volume.

Heavy Load

Finally the last step is to run a full-scale load test. This typically consumes 50 percent of the total execution/analysis time. Once the entire scenario is running, the effort shifts to analyzing the transaction response times and white-box measurements. The goal here is to determine if the system performed properly.

Typically two activities proceed in parallel — analyzing LoadRunner transaction response time data and white-box performance data. It is important to identify long LoadRunner transaction response times and internal system performance metrics that are out of range.

There is a strong relationship between this data. A long LoadRunner response time often points to a white-box performance metric out of range. Conversely, a white-box performance metric may cause a high LoadRunner transaction response time.

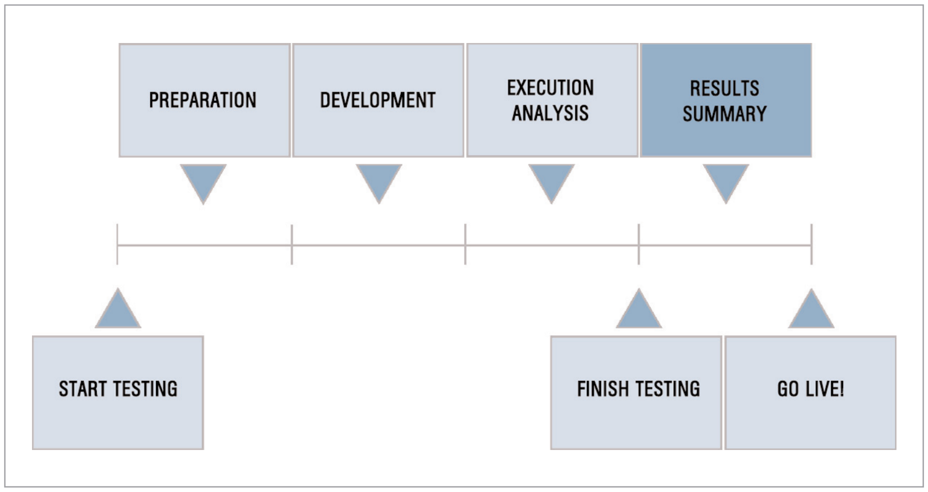
Typically white-box performance metrics help to identify areas of improvement for the system under test. The LoadRunner transaction response time will tell whether the system is functioning adequately.

It is important to note that technical staffs with administration/tuning experience are extremely valuable in this process. Individuals with database, networking, server, and application expertise are required.

From this analysis, hypotheses are generated and action items taken. The scenarios are then rerun to validate attempts to fix issue(s). A typical scenario will require more than a dozen iterations to achieve production-level quality.

Results Summary

Finally, the results summary describes the testing, analysis, discoveries, and system improvements, as well as the status of the objectives and goals. This typically occurs after the completion of testing and during any final "go live" preparation that is outside the scope of testing.



The results summary in the best practices process.

Abbreviated Best Practices Process

Often testing must be compressed into an aggressive timeframe or for upgrades to new R/3 systems that were previously load tested. This type of testing is best performed in an abbreviated best practices process. In this case, the goal should be to perform a sanity check of the system.

This strategy would be to include a representative user concurrency and transaction throughput, but with only a small number of simple business processes. Optimally the process may be compressed into four to six weeks. This process is similar to the best practices process with the following modifications:

Preparation

Analysis/Design

The design should be simplified to include concurrent users, transactions, and throughput. Considerable time savings may be achieved because this process requires little intervention from external organizations.

The following table shows this information for a particular sales and distribution module. As is evident in this table, Change Sales Order and Change Delivery do not have significant volumes to be considered as key stress test elements. In an *Abbreviated Best Practices Process* it is recommended to limit the scripting to five to 10 scripts.

MODULE NAME	NUMBER OF USERS		TRANSACTIONS, REPORTS, AND PRINT REQUESTS	EMULATE WITH LOADRUNNER	AVERAGE (TRANS. PER DAY/LINES PER TRANS.)	PEAK: (TRANS. PER DAY/LINES PER TRANS.)
	CONCURRENT	NAMED				
SD	150	175				
			Sales Orders Entry	Y	3,500/20	5,000/25
			Sales Return Processing	Y	500/10	750/15
			Change Sales Order	N	50/2	100/5
			Delivery Due List	Y	5,500/22	9,000/36
			Change Delivery	N	100/5	200/2
			Packing	Y	5,500/22	9,000/36
			Picking	Y	5,500/22	9,000/36
			Invoice Printing	Y	5,500/22	9,000/36
Total	**XXX**	**XXX**				

A sales and distribution volume estimate.

Environment Setup

There is an assumption made that the test environment is available. Therefore, the system under test should be configured and ready for testing.

Product Training

For an abbreviated LoadRunner best practices process, it is recommended to hire skilled staff so that training is not required.

Script Development

During script development, the team should develop a total of five to 10 tests. It is advisable to assign a functional individual, someone who understands the business processes, to the team for the duration of scripting. The volume data should be available. In this environment, the script development should take between one to two weeks.

Execution/Analysis

There is limited compression available to the execution phase — it will minimally require two weeks. The reduced time will compress the ability to perform system tuning. Typically the most basic tuning can be performed, although this will address only the most glaring issues.

Results Summary

The results should be positioned as preliminary — an early indicator toward eventual system performance.

Common Implementation Issues

This section describes the most common issues that surface during a load test on R/3 systems. It also suggests strategies for addressing those issues.

Preparation

Planning

The biggest issue with planning is that it is not performed adequately. Time saved is often repaid many-fold in later phases.

A good rule of thumb is to allow three months for a load test with completion at least a month before "go live." Preparation, development, execution/analysis, and results summary should each be allocated one month. Typically the results summary can be performed in parallel with late-term project tasks (data conversion/data load, user acceptance, etc.).

Analysis/Design

The first issue is to consider is, "What should I test?" and the related question "What activity does my user community create?" In many ways this is the black art of testing. Often the vendors, system architects, IT and business managers make educated guesses about the "representative user load profile." This process involves a study of the system's operational profile. It also includes an interview-analyze-negotiate process with the user community.

Business Processes

High-volume business processes/transactions should be built into the test. Choosing too few might leave gaps in the test while choosing too many will expand the script creation time. It is effective to model the most common 80 percent of the transaction throughput; trying to achieve greater accuracy is difficult and expensive. This is typically represented by 20 percent of the business processes — roughly five to 10 key business processes for each system module.

Margin for Error

Since load testing is not an exact science, there should be accommodations made to provide a margin for error in the test results. This can compensate for poor design and help avoid false positives or negatives.

A load test should include at least one “stress test” or a “peak utilization” scenario. A stress test will overdrive the system for a period of time by multiplying the load by some factor – 120 percent or greater. Peak utilization will address the testing of peak system conditions. This is often represented in such questions as: “Can the system order and ship 100,000 units/day during the week before Christmas?” “Will the system meet service-level agreements during fiscal year-end close?” “Will European nightly batch runs slow North-American online?”

Test Environment Setup

Load testing requires a complete, stable, and independent environment for testing.

Hardware

To get reliable results, the test environment must be representative of the production environment. This may be considered a significant investment; however, it offers the advantage that it can provide the facilities for ongoing testing. This is particularly useful in maintenance or system expansion. It can also facilitate a permanent testing organization.

Another option is to schedule time with a hardware vendor benchmark facility. They can provide the necessary physical hardware and hardware expertise to support a test. The disadvantage is that it needs to be repeated for each subsequent deployment.

Software

In addition to the hardware required for a load test, the test bed must also have fully installed and functioning software. Since LoadRunner functions, “just like a user,” the system would need to successfully support all user actions.

Network

Additionally the company’s network infrastructure is shared by computer systems that create a significant amount of “background noise.” While it is probably impossible to accurately model each and every network access (FTP, print, Web browse, e-mail download, etc.), it is judicious to examine the current network utilization and understand the impact of incremental network traffic.

Geography

Often the system under test will support a global enterprise. In this environment tests may often need to be run at remote sites across the WAN. WAN connectivity needs to be emulated in the lab, or assumptions must be made.

Interfaces

Large systems seldom service a company's entire information needs without interfacing to existing legacy systems. The interfaces to these external data sources need to be emulated during the test, or excluded with supporting analysis and justification.

Customization

Customization of LoadRunner is possible and has been performed in any number of environments. It typically takes considerable engineering effort and is not an "out-of-the-box" solution. Specifically customization of LoadRunner requires:

- A LoadRunner expert familiar with template programming.
- A system specialist familiar with the application API (application programming interface) and administration.
- An additional one month of project time to investigate and construct a custom LoadRunner driver, or determine if it is not possible.

Script Creation

Script creation productivity can be estimated by the "rules of thumb" from the previous section. These estimates may be strongly affected by the following issues. Any of these issues can easily double estimated script-development time.

Functional Support

One of the most important factors in script creation productivity is the amount of functional support provided – access to individuals who understand application functionality. This manifests itself when a test team member encounters a functional error while scripting – the business process won't function properly. The team member typically has to stop since he or she is not equipped with the skills to solve the issue. At that point, script creation is temporarily halted until a functional team member helps resolve the issue.

Test Design Quality

The second script-development factor is the quality of the test design. Ideally the test design should specify enough information for an individual with little or no application experience to build tests. System test documentation is often an excellent source of this information. Often designs are incorrect or incomplete. As a result, any omission will require functional support to complete script development.

Business Process Stability

To load/stress test a large system, the system's business processes first need to function properly. It is typically not effective to attempt to load test a system that won't even work for one user. This typically means that the system needs to be nearly completed.

System Changes

Finally, the last factor in script development is the frequency of system changes. For each system revision, test scripts will need to be evaluated. Each test will need to be unaffected, require simple rework, or complete reconstruction. While testing tools are engineered to minimize the effect of system change, limiting the system changes will reduce scripting time.

Test Data Load

The system will need to be loaded with development test data. This data often comes from a legacy-system conversion and will be a predecessor to the volume data for the test.

Execution/Analysis

System Changes

Often there is a shift between the systems used for development and execution/analysis. This may be created by transitions between environments onsite, or a physical move to a hardware benchmark facility.

The first step is getting the entire test to execute properly with single users. This typically involves working with each test to account for system changes. Difference in data load, configuration, system version, etc. can cause issues. It is recommended to perform a backup of the development environment that may be restored for execution. This should help to avoid version and configuration issues.

This analysis is especially important in global operations where one continent's batch processing is running concurrently with another continent's online processing.

Data Seeding

Often business processes consume data (e.g., a shipping request requires a sales order that, once shipped may not be shipped again), and their tests will also consume data. Because of this the system should be "pre-seeded" with data consumed by the testing process. To keep testing productivity high, there should be enough data to support several iterations before requiring a system refresh – which should be done each night.

Volume Data

System performance may vary widely with the volume of system data. The system under test should have sufficient data to model the size of the product database at the time of deployment. It is also judicious to expand the database size to model the system performance with an additional one to two years of data in the system.

System Support

Just as the development requires functional support, execution/analysis requires system support because the test team may not understand the system architecture and technology.

The purpose of system support is to help interpret LoadRunner results and white-box data. While LoadRunner will describe what occurred, the system support can help to describe why and suggest how to remedy the problems. These suggestions can be implemented and the tests rerun.

This iterative process is a natural part of the development process, just like debugging.

Summary

Corporations with planned long-term reliance on R/3 systems are learning too late about the costly consequences associated with failing to align a continuous and repeatable load testing strategy with their strategic R/3 implementations. Employing repeatable load testing practices upfront eliminates IT from having to take the fire-drill approach to addressing such problems as poor performance, system incompatibilities, or complex business process reengineering near or into production. The business benefit for utilizing the best practices for load testing described in this document, whether using the full or abbreviated best practices process, will result in higher performance, less application re-work, less maintenance, and less downtime cost in production.

Download a 10-day trial of Mercury LoadRunner® and see how load testing your applications in pre-production will save you time and money when you're ready to go live.
http://download.mercuryinteractive.com/cgi-bin/portal/download/loginForm.jsp?id=160&source=1-102709805#d5160