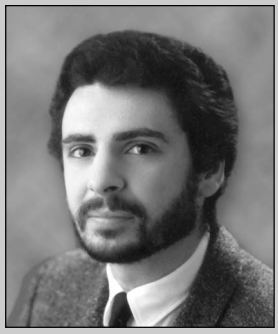
# A Custom Performance Test — Is It for You?

Susanne Janssen and Ulrich Marquard

*Susanne Janssen, co-developer of SAP's Quick Sizer tool, holds responsibility for information development for sizing, benchmarks, and performance at SAP AG.*



*Dr. Ulrich Marquard, Director of SAP's Performance & Benchmark group, holds responsibility for system architecture, analysis, scalability, and performance at SAP AG.*

*(complete bios appear on page 24)*

When installing or planning new systems, or when migrating to R/3 from R/2, customers often come to the Performance & Benchmark Group asking for our assistance in devising a customized performance or load test.[1] Their objective is to test the viability of their newly designed hardware system to make certain that it will adequately perform when supporting the real-world requirements of their users and business processes.

Our response often comes as a surprise. "Do you *really* need to conduct a load test?" We ask this because we believe that the overwhelming majority of customers who install new systems do not need to devise and conduct a resource-intensive load test. To model their systems and accurately gauge the system's characteristics, they can use the SAP Quick Sizer tool — which is available for customers, partners, and prospects with a customer number on SAPNet at **http://sapnet.sap.com/quicksizing** — and SAP's Standard Application Benchmarks (or results from standard application benchmark tests run by their hardware vendors, which are posted on **http://sapnet.sap.com/benchmark**).

When *does* it make sense to conduct a load test? It makes sense — in fact, it's an absolute *must* — for large, complex, or highly customized systems with add-ons. In these situations, a load test, whereby you physically test and measure a heavily loaded system before it is deployed, is the only way to verify that the hardware sizing, configuration, and parameterization of your system will adequately

[1] Otherwise known as a "stress test" or a "multiuser load test."

support your business transactions. In other words, the load test is the only way to tell if the system you have configured will deliver the response times and throughput you are expecting.[2]

How do you know if your system is one that requires a custom load test? We'll show you how you can answer this question for yourself in the first part of this article. Then, suppose yours is indeed an SAP system that does warrant a custom load test. How do you proceed? What steps must you take to successfully devise and then execute a custom load test? This article answers these questions as well.

## *Does Your New SAP System Require a Custom Load Test?*

We suggest that the following SAP systems undergo custom load testing before going live to ensure that the business processes, the interfaces, and the overall system, including network load and disk I/O, perform as expected:

✔ An SAP system that is large — i.e., one that goes beyond the CPU or disk limits of the Quick Sizer (for example, one that starts off with a disk size of more than 400 gigabytes).

✔ An SAP system with very high throughput requirements for certain processes. For example, there have been customer performance tests for processing more than 1 million POS Inbound line items per hour, or 1 million Financial documents per hour.

✔ An SAP system that is subject to very demanding time constraints, such as a tele-sales company where users need very short response times.

✔ A system landscape that is complex (e.g., a large and intricate network that is spread across the globe with systems in Silicon Valley, East Asia, and Europe, and uses satellite connections).

✔ An SAP system with numerous custom interfaces or add-ons where the customer has added their own coding or third-party products. Of course, the behavior of this coding cannot be predicted by SAP and therefore requires a deeper performance analysis than what is outlined here in this article.

### Situations Where a Custom Load Test Is Not Required

✔ **You are using an AcceleratedSAP (ASAP) Ready-to-Run SAP system.** For this small- to medium-sized, preconfigured Basis system, you can use the proven ASAP Roadmap methodology and standardized project planning to implement your SAP system and address the technical issues you are likely to face (such as your interfaces and data conversions). A Ready-to-Run system enables you to bypass the custom load test and get right to the technical implementation.

✔ **You are using the standard SAP functionality without any custom add-ons, interfaces, etc.** The standard process and SAP system scenarios have been well tested (often using benchmarks) for their scalability.

✔ **You have only a small number of custom add-ons, interfaces, or reporting functions.** Even with some custom add-on functions, you may not need to conduct a custom load test, although you still will need to take certain precautions, such as conducting a careful performance analysis of these functions.

---

[2] Additionally, you may opt to use performance tests to:

- ✓ Prove the efficiency of your system landscape (including the LAN and WAN connections) in which the SAP system will operate — this includes testing the interfaces to other SAP or legacy systems.
- ✓ Verify strategies for backup, recovery, and high availability.
- ✓ Verify strategies for data security.

## A 13-Step, Two-System Testing Methodology

In this article, we assume that you already have the following three systems available: a development system for customizing your own code; a quality assurance system for testing and consolidation; and a production system, which is the system the custom load test is for. We recommend a 13-step process for devising, executing, and interpreting the results of the custom load test.

Before we walk you through these steps, let's take a moment to discuss the underlying, two-system test methodology around which these steps have been designed. Load tests are carried out on two systems[3]:

- **Development system** — In this system you conduct initial performance tests in single-user mode. It should support on-the-fly changes to code during your performance analysis. In most cases, it is the small in-house system you use for developing code.

- **Load test system** — This is the system where you actually run the load test and simulate your real-world, multiuser conditions. It doesn't matter whether this system is on the machines of the quality assurance system or the future production system, because after the test, the data will be deleted. What we do recommend is that the load test system be identical in infrastructure, layout, and tuning to what will be in the production system.

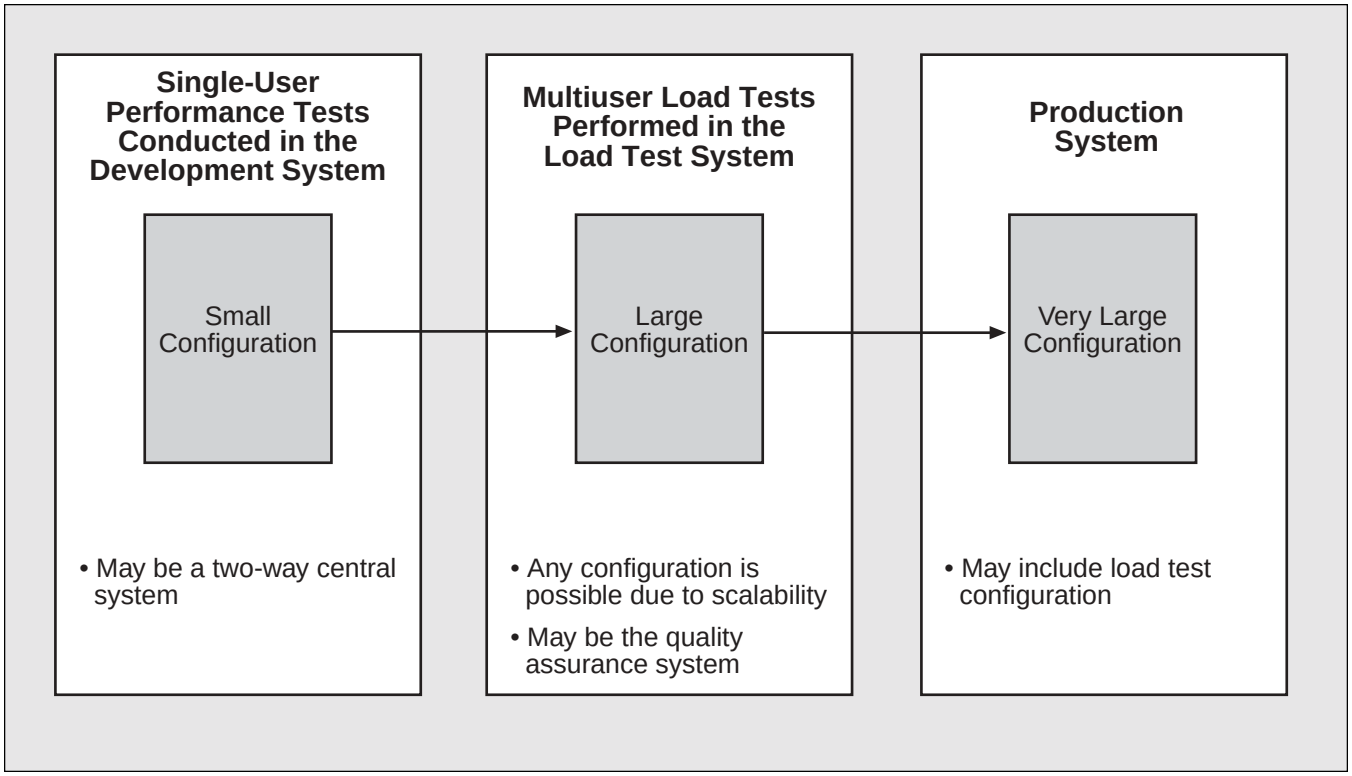
**✔ Tip**

*The performance optimizations you make in single-user mode in the development system enable you to estimate the database calls, the database size, and the CPU and memory consumption needed to implement and run a particular business process. This pre-test process also gives you a solid basis to judge which processes you want to test under load in a multiuser environment, and those that you do not. Choosing your test scope wisely will save you time, energy, and money. That load test system is typically an expensive, large, dedicated system that requires setup by specialists, internal or external. In addition to its setup, this system needs to be configured with data that accurately models your real-world environment. This is a costly, time-consuming process, which you need to get right. The scalability of your system depends heavily on the data distribution that is carried out across the test system. For example, the computing requirements of an SAP system that needs to support thousands of sales orders with the same material for the same customer differ greatly from the requirements needed to support thousands of materials and customers.*

Don't forget that performance analysis is an iterative process, whereby processes are tested and optimized — first, in single-user mode on the development system; then, concurrently in multiuser mode on the load test system. On a small development system, changes in the customizing data, the master data, ABAP code, Basis configuration, operating system, and database parameters can be made with relatively little cost and effort. Once all these things have been refined to meet your expectations, you then test, analyze, and optimize the business processes concurrently on the load test system. Here, in

[3] Please note here that we do not necessarily want you to buy new hardware. Knowing that hardware costs are settled at an early stage, we recommend that you make use of what you already have. Therefore, when we refer to different systems in the article — e.g., load test system, quality assurance system — the difference between them is *logical*, not physical, meaning that two different systems may run on the same physical machines. For example, one of our customers already had a production system and wanted to add new functionality. However, he wasn't sure whether the software could handle the particular volume, so he bought a new machine and ran a load test for the new functionality. When the test was successfully completed, he deleted all the data and added the new machine to the production system to serve as an application server. That way, he could efficiently reuse the hardware. The new machine served as both a load test system and a production system.

*Figure 1 System Landscape for a Load Test*




multiuser mode, you run tests to validate the scalability of your SAP system — including system infrastructure, such as network or disk I/O — and performance forecasts. When you are not pleased with your findings, or if you discover that a prediction made regarding the multiuser tests proves to be off base, you can return to the single-user development system to optimize an individual process. The testing system landscape is shown graphically in **Figure 1**.

## The 13 Steps of a Custom Load Test

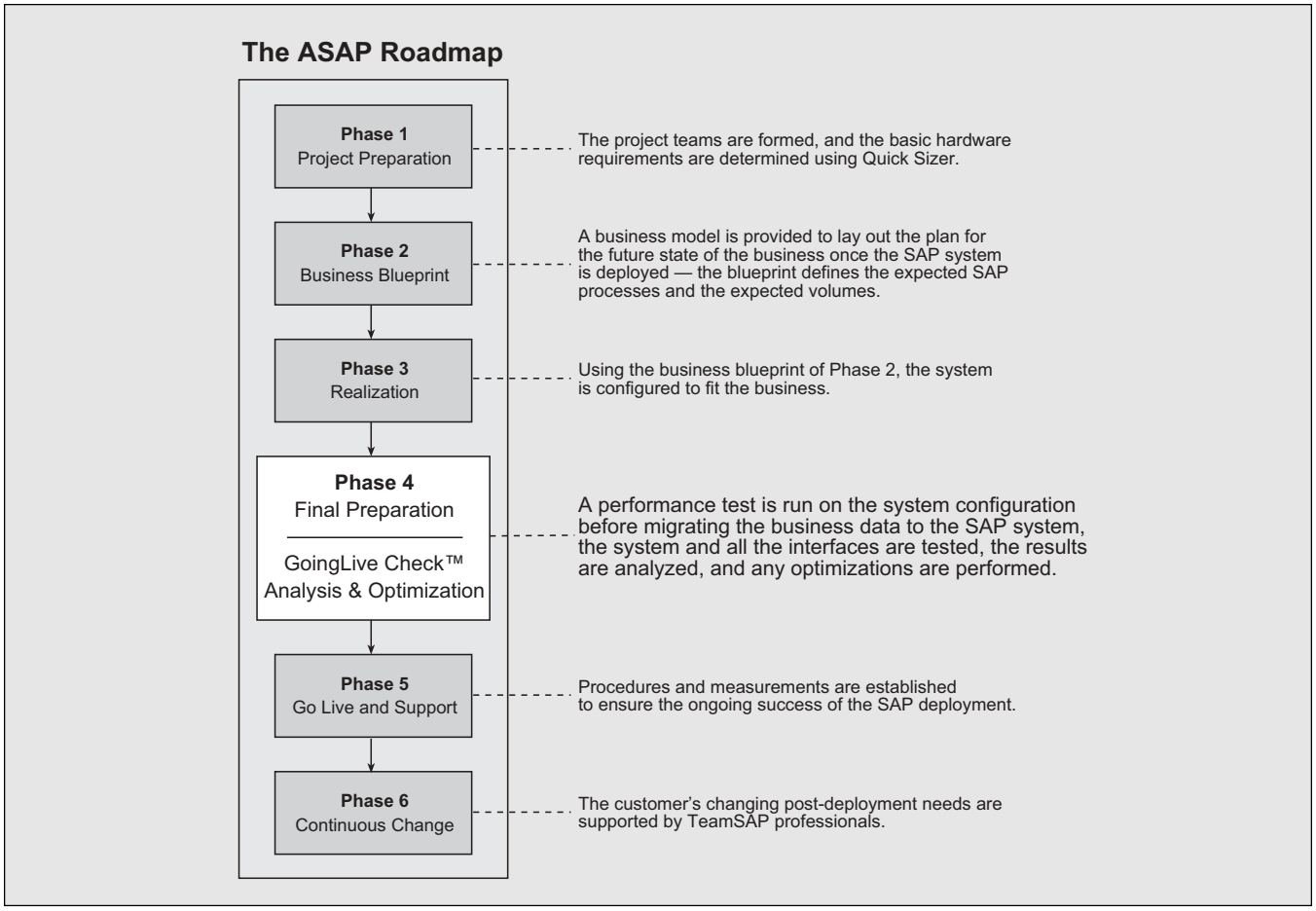
Listed below are the 13 steps you'll need to follow in order to devise, execute, and interpret the results of a custom load test[4]:

1. Determine which critical business processes need to be tested.
2. Determine the expected user and system response times and throughput figures.
3. Determine the functionality needed for the load test — for example, ATP (Available-to-Promise) Check, pricing, workflow, or interfaces to other systems.
4. Determine the data distribution — i.e., the type of master data you need.
5. Establish test scenarios for the business applications.
6. Perform the initial capacity planning for the load test system.
7. Devise test scenarios for the test systems.
8. Determine the load test system infrastructure.
9. Determine the methods of load and user simulation.

[4] Note that steps 1 through 9 are planning activities. You don't actually begin work with the development and load test systems until step 10 and step 12, respectively.

*Figure 2 The Load Test Is Run During the Final Preparation Phase of the ASAP Roadmap*




10. Create the actual test data. (Steps 10 and 11 are performed using the development system.)
11. Run the single-user load tests, and measure and analyze the performance.
12. Run the multiuser load tests, and measure and analyze the performance. (This step is performed using the load test system.)
13. Evaluate the test results.

## Paving the Way for a Successful and Effective Load Test

We recommend that you conduct your load test as part of the GoingLive Check™ within the AcceleratedSAP (ASAP) program.[5] The usual procedure is to conduct the load test during the Final Preparation phase of the ASAP Roadmap (see **Figure 2**). The preparation of the test, however, goes hand-in-hand with the individual phases of the Roadmap.

The object of a load test is to test all the mission-critical business processes in the SAP system

[5] ASAP is the comprehensive TeamSAP implementation solution used to streamline and standardize the life cycle of the SAP system. The ASAP Roadmap provides customers with step-by-step guidance throughout the entire SAP implementation process, helping to optimize quality, timelines, and resource usage. The GoingLive Check™ is a TeamSAP service, in which SAP specialists analyze your SAP system, inspect the configuration of individual system components, perform the Sizing Plausibility Check, and give you recommendations for optimizing your system before the start of production.

for expected user/system response times and system throughput. As such, load testing can be a huge undertaking — resource-intensive in terms of people, time, hardware, software, and, of course, money. You will need to establish project teams to make effective use of these resources and to devise, execute, and evaluate an effective load test.

### *Forming the Project Teams*

Behind every successful load test is a team of individuals who represent all the areas affected by the SAP system. These individuals have various responsibilities for preparing and conducting the load tests. They define the most critical business processes and some typical scenarios for these processes. They determine the volumes of data that have to be processed. They specify time frames. They create the load test system, ensuring that the system is sized to meet peak time requirements, and conduct detailed analyses of the performance results. If the results do not measure up to expectations, the team attempts to find out why.[6]

So, the first order of business is appointing a project leader to head up this diverse team. The project leader will be responsible for planning the human resources and securing the hardware and software resources for the load test, and for the installation and the point of realization of the interfaces and customer add-ons. Don't move forward without this job function in place. Having someone with an overview of *all* activities is absolutely critical to the success of the load test.

Once the project leader is in place, you should establish three project teams:

- An Implementation team to define the load test objectives
- An Application team to check the application behavior and tune the application
- An IT team to check the system behavior and tune the system

The **Implementation team** will establish the objectives of the tests, and should ensure that the deployment of your SAP system implementation or upgrade follows SAP's AcceleratedSAP (ASAP) methodology and GoingLive Check™, which can help you recognize and resolve performance problems.

You want the Implementation team to broadly represent all the groups that you expect will be affected by the performance of your SAP system — i.e., it should include people from a whole variety of backgrounds: actual users of various applications, various product managers and management heads, and various IT and application specialists. This broad representation is more important than the size of the team. In fact, the size of the Implementation team can vary greatly, depending on your particular needs. We know a customer, for example, who successfully executed a load test with a rather large Implementation team consisting of 57 members. On the other hand, we've seen a team that consisted of just two specialists from each tested application component (six in this case), and two IT specialists.

Once the Implementation team is in place, you set up the **Application team**, which will work on the implementation of the load tests from the point of view of the business applications. Members of this team will also sit on the Implementation team. Ideally, this team should be kept small — minimally, it could consist of just two or three members: a senior program analyst, a functional analyst, and a senior programmer, for example. In most cases, the Application team can make use of the results of Phases 2 and 3 of the ASAP Roadmap, shown earlier in Figure 2.

Next, you form the **IT team** with IT specialists who will verify that your load test system is well laid

[6] At this point, the project team may call in SAP performance specialists to detect the culprit and improve the performance of the entire process.

*Figure 3* *Project Team Tasks*

| Task | Project Team | | |
|---|---|---|---|
| | Implementation Team | Application Team | IT Team |
| 1. Determine which critical business processes need to be tested. | ✔ | | |
| 2. Determine the expected user and system response times and throughput figures. | ✔ | | |
| 3. Determine the functionality needed for the load test. | | ✔ | ✔ |
| 4. Determine the data distribution. | | ✔ | |
| 5. Establish test scenarios for the business applications. | ✔ | ✔ | |
| 6. Perform the initial capacity planning for the load test system. | | ✔ | ✔ |
| 7. Devise test scenarios for the test systems. | | | ✔ |
| 8. Determine the load test system infrastructure. | | | ✔ |
| 9. Determine the methods of load and user simulation. | | ✔ | ✔ |
| 10. Create the actual test data. | ✔ | ✔ | ✔ |
| 11. Run the single-user load tests, and measure and analyze the performance. | | ✔ | ✔ |
| 12. Run the multiuser load tests, and measure and analyze the performance. | | ✔ | ✔ |
| 13. Evaluate the test results. | ✔ | ✔ | ✔ |

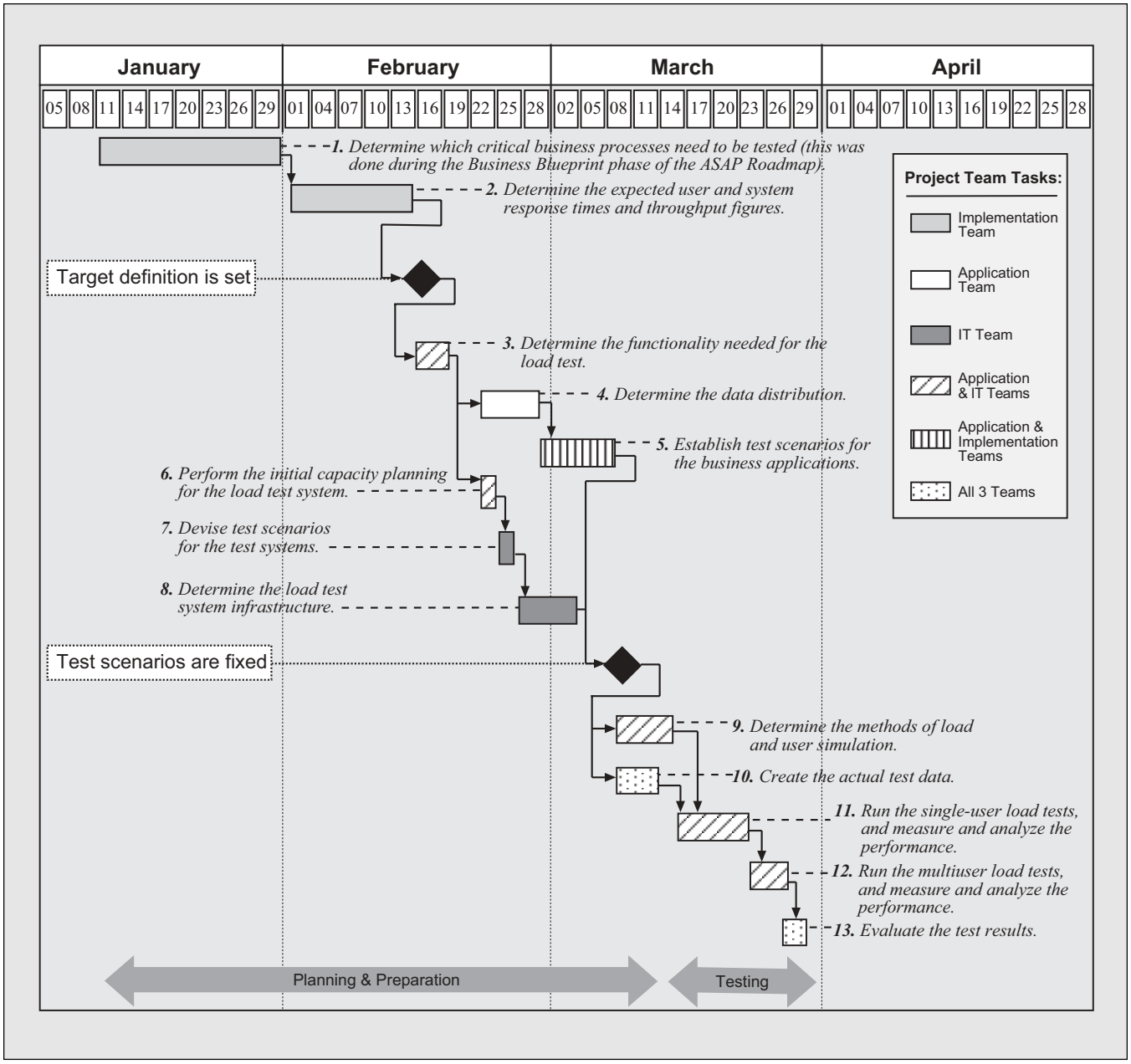
out — i.e., they'll make sure that the existing hardware and software infrastructure fits together well with the SAP infrastructure.

This team will also oversee most of the system performance analysis work, and should include systems experts and database specialists. So, it should also include one or more SAP systems analysts[7] with expertise in the system landscape and infrastructure, as well as senior systems analysts and individuals from database administration who have expertise in the areas of database layout, operating systems, database systems, network technology, etc. The IT team members will also sit on the Implementation team.

**Figure 3** shows how these three teams support the 13-step load testing process we outlined earlier.

[7] Your SAP systems analysts may be from TeamSAP, representatives from the hardware vendor organization, and/or outside consultants.

*Figure 4 Sample Load Test Preparation Timeline*




The sample timeline depicted in **Figure 4** details how these activities can be scheduled among the three project teams. Note that the time allocated for the various team members can be anywhere from 50 to 100 percent, and that the entire load testing process can take up to 2.5 months in the context of large projects where the processes being tested are sizeable and complete. While it is possible to conduct an entire load test over the course of one night, in the situations we have seen this, the processes being tested were very small and lean. (Timing, of course, depends heavily on the number of critical business processes and the time needed for Phases 2 and 3 of the ASAP Roadmap.)

In the following sections, we'll take a look at each of these activities. Before we proceed, however, let us offer you two pieces of advice:

✔ **Don't reinvent the wheel if you don't have to!** If you know of another organization that has an SAP system similar in nature to yours and has executed a customized load test, see if they will share their experiences and test plans with you. It can save you countless hours and dollars!

✔ **Document all your efforts!** We cannot overemphasize the importance of creating detailed documentation for each of the 13 steps mentioned earlier. Documenting the steps you took and the decisions you made is essential for tracing errors, and for knowledge transfer when new people come onboard. It is also important to document experiences such as setting certain parameters during the tuning and optimization phase.

Imagine, for example, you are in the middle of tuning the load test system, and you incidentally discover that a certain profile parameter setting in combination with a little switch doubles the speed of the business process. Of course, you quickly make the necessary alterations and move on to the next task. Without documentation of these changes, however, you or a colleague may end up trying to solve the same problem again during the preparation of the production system, wasting valuable time and money. To avoid a situation like this, it is a good idea to have comprehensive documentation of every step taken.

*We cannot overemphasize the importance of creating detailed documentation for each of the 13 steps mentioned earlier. Documenting the steps you took and the decisions you made is essential for tracing errors, and for knowledge transfer when new people come onboard. It is also important to document experiences such as setting certain parameters.*

## Step 1: Determine Which Critical Business Processes Need to Be Tested

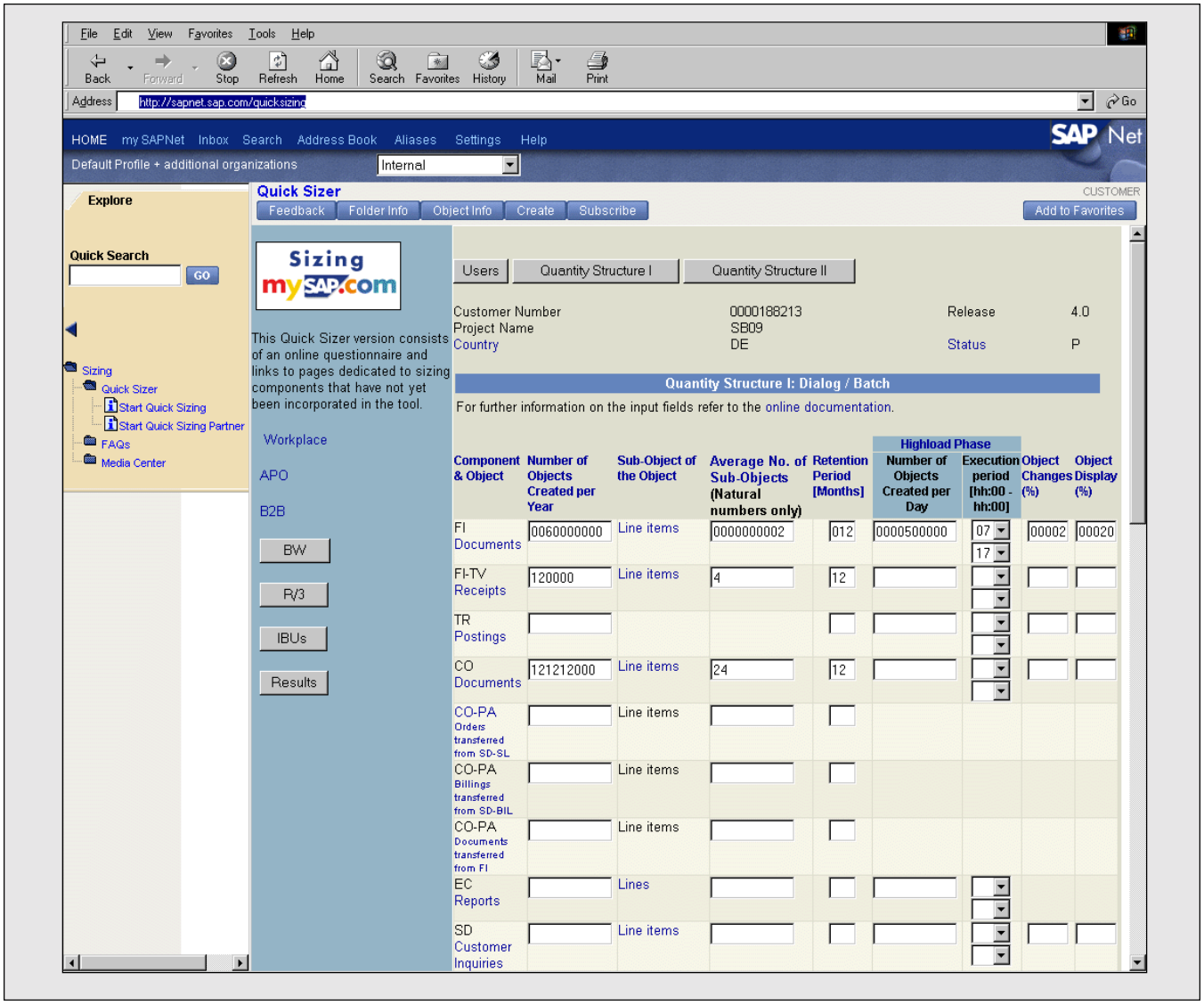
If you've run through Phase 1 (Project Preparation) and Phase 2 (Business Blueprint) of the ASAP Roadmap, steps 1 and 2 really shouldn't take much time, because most organizations will have already determined their two to three most critical business processes. In the project plan shown in Figure 4, we assumed that Phase 1 and Phase 2 still had to be conducted.

In step 1, we basically subdivide between testing the response times of particular business processes (e.g., a tele-sales rep dealing with hundreds of orders per day would need a two-second response time for creating a sales order in the system, while for others the response time covers the time between the creation of a billing document and the printout of the invoice), and business processes that either run in parallel or succeed one another (e.g., in a particular SAP Retail system, the POS Upload has to be finished by 7 PM so that the replenishment run can take place before midnight). Another example for high throughput, and thus a critical process, may be a month-end or a year-end processing. These critical business processes will have been defined in Phase 2 of the ASAP Roadmap.

## Step 2: Determine the Expected User and System Response Times and Throughput Figures

Once you have detailed the critical business processes for the test, you write down the numbers you are expecting. For example, let's say that you want to test whether or not the system can handle 3 million POS Inbound in 1 hour, or whether the system has an average response time below 2 seconds when 10,000 users concurrently create sales orders with numerous line items. To determine these figures, you can draw on experience from your current business.

*Figure 5 The Quick Sizer's Sample List of Business Processes*




After this step, you should have a list of the business processes, the respective response times, and the throughput figures. If you need a model to start from, you can use the Quick Sizer's list of the most important business processes per application. This is shown in **Figure 5**.

With this list in hand, you can start assessing the overall costs of the load test. Obviously, the longer the test, the higher the costs. To predict these costs with any degree of accuracy, your schedule needs to accurately reflect the workdays and available time of the various team members involved.

Generally, the amount of time you'll spend preparing for and implementing a load test will largely depend on two factors: the load you are going to produce and the complexity of your business processes. For example, it will make a big difference whether you will be testing batch jobs or users (either simulated or real users) — i.e., the simulation of users with simulation software requires more effort than scheduling batch jobs.

There is a difference between sales orders created by tele-sales users in a call center and sales orders that are imported via an interface to the system and

*Figure 6* *The Dialog Steps of an SD Benchmark*

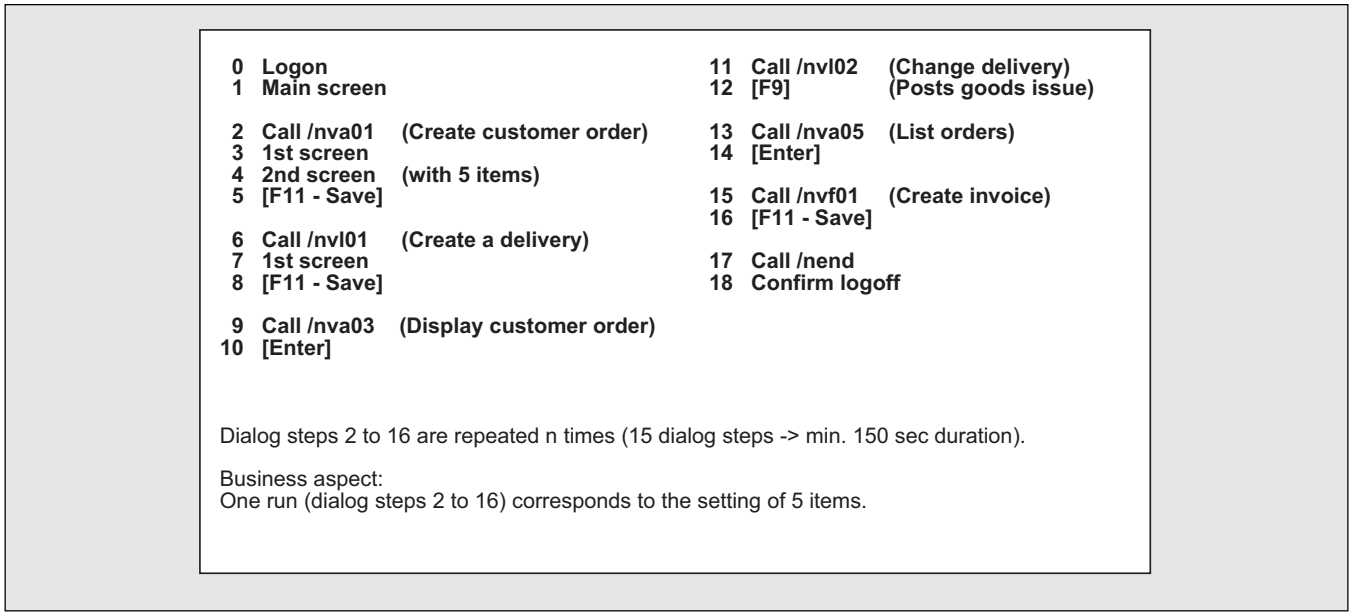
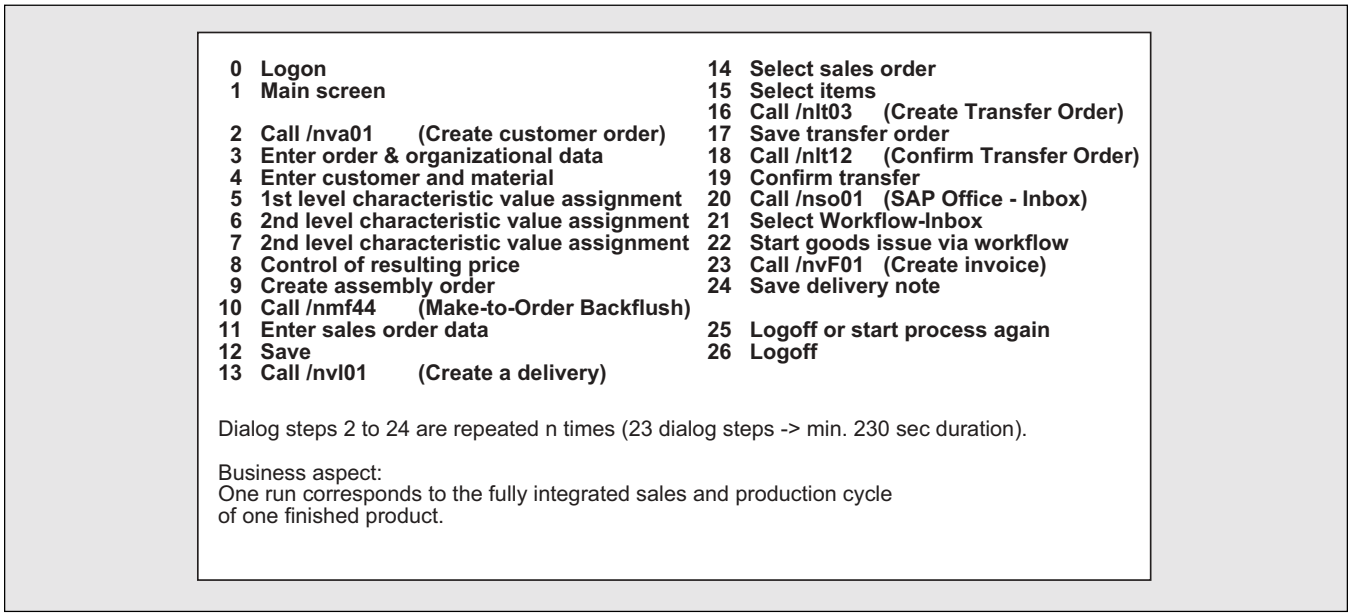



*Figure 7* *The Dialog Steps of an ATO Benchmark*




therefore can run in batch. In the first case, the business process of creating a sales order is run through completely step by step. In the latter case, the entire sales order is being uploaded in the background. Also, testing a rather straightforward business process (such as creating a purchase order requisition) will take less time than testing an ATO (Assemble-to-Order) scenario that includes product costing, workflow, invoicing, etc.

To give you an idea, we have included the business scenarios for an SD standard application benchmark (**Figure 6**) and an ATO benchmark (**Figure 7**).

## Achieving a "Leaner" Load Testing Process

As a precondition for "lean" load testing, you need to ensure the success of two project activities:

- Determining and implementing the critical business processes
- Creating the master data

To save time in completing these two tasks, you can import the benchmark client or the IDES client as a starting point that contains a set of business processes and also master data. Another possibility is to use preconfigured business processes called Business Configuration (BC) Sets, available from SAP. In our experience, once your business processes and master data are all set, you can usually expect the actual test to take another two weeks. A longer time frame suggests that there were mistakes in either of the two project activities. This is to say that you should make your life easier and take advantage of data that is already available in the system.

## *Step 3: Determine the Functionality Needed for the Load Test*

After steps 1 and 2, you have probably settled on four critical business processes — e.g., POS Upload, Replenishment, Sales Order Creation, and year-end processing. In step 3, the Application and IT teams work in close cooperation, and must settle on what is realistic or possible to test, with a view that's tilted toward the IT perspective. For example, while the Application team might want to test the year-end closing process, the IT team may find out that importing the data from another system could take too long. In this case, the two teams may decide to use only 50 percent of the system resources and test only 50 percent of the volume. They could then scale the business process high — i.e., double the result obtained.

While the teams are determining the scope of the test, they may at the same time settle on the processes that are best suited for a single-user analysis in the development system, and those best tested in a multiuser environment in the load test system. For example, the costs for simulating the work of an entire day are very high. Some of the processes that take place in a day's work are not significant for performance, but still have to be incorporated into the load test script. This might be too much for little efficiency. Naturally, the processes for a full-size test and those for a smaller-volume test will have different expected runtimes or throughput.

Determining the load test's scope also means deciding on whether you want to test particular processes, or test the processes of an entire working day, or whether you want to test disaster recovery to find out how long the system needs to back up and recover from media failure. Each kind of testing will take a different toll on the system.

It is important to know what exactly you are going to measure and how far you want to go. By the end of this step, both teams need to sit together and try to answer the following questions:

- Which processes are worth being tested? In step 5, the Application team will work out the details of setting up the necessary high-level test scenarios.

- Will infrastructure tests be conducted? In step 7, the IT team will detail how they plan to proceed with such tests.

- Will the processes be tested on a 100 percent basis, or less due to SAP's scalability — i.e., will you test whether the system can handle 200,000 FI documents in 1 hour, or will you restrict yourself to 100,000 FI documents per hour? Of course, you mustn't forget about this when you do your final sizing, otherwise the production system will be too small.

A few years ago, an FI customer conducted a huge load test. They set up a large system and ran a few FI business processes for a couple of hours until

the system performance broke down. The system was maxed out because the main business process was incompletely modeled. As a result of the incomplete model, the customer created millions of financial documents without balancing them. Part of the test was that all simulated users displayed all open, unbalanced items. In a real-world scenario, a company with millions of open postings would face thousands of unpaid bills and thus would be close to bankruptcy. So our parting advice for this step is to make sure you model testing scenarios on real-world scenarios.

## Step 4: Determine the Data Distribution

The Application team defines the data basis that is going to be used for the test. This step involves clarifying where the data for the test will come from, including how (which methods will be used) to create the master and customizing data. They also develop a well-defined data design that will ensure that the results can be reproduced.

When determining how to distribute the data that will serve as the basis for the test, be very careful not to create artificial bottlenecks. For example, you should avoid doing all the work on only one cost center, or one customer. And you should make sure that the data that is created during multiple tests will have no impact on further testing — so, when creating and displaying open items, be sure to balance them. Recently, we were called in to help a customer who created a data distribution bottleneck when testing invoicing. Unfortunately, all their invoices were running on the same payer. Although they had different "bill-tos," with only one payer, the bottleneck that they created resulted in extremely long waiting times. You can avoid this situation (also known as the "cloning effect") if you do not create vast amounts of data by simply making identical copies of it. Thus, along with the complexity of the business process, for your master data, you may need to create 5 to 10 different sets of master data.

It is also very important to think about probabilities of certain attribute combinations, such as how many different customers will order how many different articles per day. This has a big impact on table sizes as well as response times for reporting for the SAP Business Information Warehouse (SAP BW) processes. Some business processes, for example, can be repeated over and over again (such as the process for creating a sales order) because they are independent from the current state of the system, whereas other processes (such as billing) require a certain foundation of data.

There are several techniques you can use to get your data for the test. For example, you can:

- Use data migration tools such as batch-input, BAPIs, and IDocs. The advantage with these is that you can rely on your own data, and at the same time you can test importing and migrating data.

- Use simulation tools such as SAP's Computer Aided Test Tool (CATT) to generate and manage the master data, if you do not use real-world data. An advantage of this method is that you can perform functional tests with it, too. You can run the test on the same data so that the test results are reproducible.

- Use the SAP Standard Application Benchmarks, which contain a tool for simulating dialog users. Before using the benchmarks, however, we suggest you contact an SAP consultant who is experienced with their use because they were written for technicians.

For master data creation, we recommend you try the first method. On one hand, you can rely on your own data; on the other hand, you can test data import, which can also be critical for performance.

In any case, if you are under time constraints, we recommend you choose the method that is the most straightforward for you to use.

✔ **Tip**

*Whichever method you use to get your data, it is important to determine how to save and reset the data once it has been used. For this we recommend you restore the database or re-import the test client. Restoring the database is usually faster, and also provides a good training opportunity, and a good way to see how fast this works. On the other hand, re-importing the test client with a database-independent transport utility from SAP is more flexible because you can easily transport the data to a different platform.*

## Step 5: Establish Test Scenarios for the Business Applications

In this step, the Application and Implementation teams proceed to set up detailed test scenarios, in keeping with the order of the business processes being tested. For a Retail scenario, for example, they would start by setting up *POS Upload*, followed by *Replenishment*, then *Ordering*, and finally *Goods Issue*.

Here, you basically follow the model shown earlier with the benchmark descriptions for SD and ATO (see Figures 6 and 7). You carefully establish, step by step, how the business process is mapped to the system — i.e., logon, first screen, enter a particular data in a particular field, choose the follow-on screen, etc.

We did this using the SAP Standard Application Benchmarks. In Figure 6, we detailed the steps that we worked through. We wrote down:

- The transactions that were run through
- The sequence of the transactions
- What data needed to be entered where

There are also other techniques you can use to illustrate the test scenario. For example, you could capture screen shots of the various screens and pin them to the wall, or you could also use paper mockups.

By the end of this step, the business process being tested should be run through and detailed completely, and all dependencies should have been identified and considered. Keep in mind that people may have different definitions of "complete." One person might consider a sales scenario complete only when the goods have been moved to the truck and the billing has taken place, while someone else may want to omit warehouse management from the sales process because they sell services. It is important to make sure that everyone is on the same page.

## Step 6: Perform the Initial Capacity Planning for the Load Test System

In this step, the Application and IT teams together conduct the capacity planning for the load test system by sizing it and laying it out.[8] The teams can use the Quick Sizer to get an initial idea about the size needed for the hardware. In most cases, the hardware vendors are also involved at this point, since they are responsible for sizing in the end.

Keep in mind that the exercise of sizing the load test system offers a good training opportunity for the eventual sizing of the production system.[9]

## Step 7: Devise Test Scenarios for the Test Systems

Here, the IT team creates detailed test scenarios for the hardware configuration. Just as with the test

[8] For a detailed look at what's involved in capacity planning, refer to Kurt Bishop's article, "Size Does Matter — Strategies for Successful SAP R/3 Capacity Planning," in the Premiere Issue of the *SAP Professional Journal*.

[9] Note that step 6 is conducted concurrently with step 4. The timeline you saw in Figure 4 shows that once you determine the test scope in step 3, you can undertake both the data distribution in step 4 (the Application team) and the capacity planning in step 6 (the Application and IT teams).

scenarios for the business applications (which were prepared by the Application team in step 5), the IT team sets up these tests in keeping with the order of the business processes being tested.

For example, let's say the teams decided to test the WAN connection. In this step, the IT team would determine how to establish a WAN infrastructure that matches the requirements derived from the business processes.

## Step 8: Determine the Load Test System Infrastructure

The IT team now prepares the test environment by setting up the system infrastructure — i.e., the number of servers, the number of GUIs, the network, the disk layout, the number of work processes per server, etc. — for the load test system.

As mentioned earlier, we assume in this article that a development system for the single-user tests was already available when this testing process began — just make sure your tests can run smoothly by ensuring that the system is not modified during the tests and that it is bug-free.

While you can install the hardware for your load test system at any time, it may make sense for you to hold off until you've measured your single-user performance on the development system, particularly if your new SAP system landscape is very large or distributed. In these cases, you will find the results of your single-user tests can help you size your load test system and thus your production system. So, we recommend that you wait to set up the load test system until *after* you've assessed your single-user measurements — which will be sometime during step 11. This approach has two benefits:

- First, it enables you to take advantage of your single-user test measurements to more accurately estimate how much hardware you'll need to scale your load test system for the multiuser load tests.
- Second, it allows you to set up the equipment right before it is needed. Depending on the level of detail of your single-user performance analysis and optimizations, it may take some time before you will be ready to actually run your multiuser load tests. In the interim, you can avoid tying up the hardware you set up for the load test system — which is likely to include some rather expensive resources.

Note that most hardware vendors are interested in partnering with customers in the load test. This means that you should contact your hardware vendors for assistance.

### *Helpful Hints on Creating the Load Test Systems*

✔ To avoid overloading and bottlenecks, make sure that when you do the database layout, hotspot tables are not located on the same disk.

✔ If you run a complete test of a critical central business process that is closely connected with an interface, you have to incorporate the interface into the test as well — at least enough to allow you to run the test. For example, if you have a tax calculation interface, you must set this up and run it, too. And if your production system is to include a WAN connection, you should test the WAN connection rather than the LAN connection.

✔ It's a good idea to prepare backup and restore strategies for the two test systems — the development system and the load test system. Even the small development system ought to have regular backups so that you can repeat a test phase or start a test from scratch again. Keep in mind that load testing is an iterative process — during the test, you are likely to come across problems, and you might have to go back and start a phase again.

We recommend you perform backups in addition to documenting your efforts as you go along — for

example, anytime you make a change to a program, to the customizing settings, to the master data, or to the database layout.[10]

✔ You can prepare your multiuser load test system quite easily by using a two-team approach, as you'll see shortly in the section *Tuning the Load Test System*.

## Step 9: Determine the Methods of Load and User Simulation

Here, the Application and IT teams determine how the load will be produced — i.e., whether the data will be entered via dialogs, batch processing, or by interfaces.

For dialogs, the IT team (who will be doing the actual implementation) may choose to have the data input by real users, or generated by tools such as the SAP Standard Application Benchmarks that can simulate user inputs. The main advantage of having real users input the data rather than simulation tools is that working with the system gives them an opportunity to get a look and feel of the SAP system. Also, in return, the users can help to ensure the quality by working through the functionality of the business process.

However, if you do decide to test with end users, don't fall prey to assuming that they know how to proceed on their own. If not given very detailed step-by-step procedures, an end user might put the system under load by executing very complex business processes that have nothing to do with their actual work, just to "stress" the system. A step-by-step procedure not only keeps end users focused, it also includes time scenarios that reflect the actual workload end users have to cope with.

[10] To back up the customizing settings and the master data, we suggest you export the client or create a new client.

## Step 10: Create the Actual Test Data

All three teams — Implementation, Application, and IT — are involved in importing or creating the data that will be used in the load test. Together, they define the master data and customizing settings. Remember the example of the FI customer with the bill-to scenario? In a single-user scenario, you need just one bill-to. In the multiuser load test, however, you need many more.

This step requires real hands-on work from all of the team members involved. If you did your planning and documentation well, however, it should be a fairly easy job.

## Step 11: Run the Single-User Load Tests, and Measure and Analyze the Performance

You begin the load test by running a series of single-user performance analyses on the development system, which means testing a single process at a time. According to your findings, you can apply any bug fixes and performance optimizations, and repeat the tests until you are satisfied with the results. At this point, you'll also make predictions about how you expect the load test system, and henceforth the production system, to perform under load — i.e., what happens in the next step, where you move to the load test system and run multiuser load tests on concurrent processes. If these forecasts are not met, you'll return to this step and repeat the single-user performance analysis on the development system.

Note that for the single-user tests, you do not need actual users. In fact, you really only need one or two performance specialists who can analyze the business processes, measure the CPU times, etc. Then, relying on the scalability of SAP, you can

*Figure 8 Strategy for SAP Performance Analysis*

| Test Findings | Objective |
|---|---|
| Single-user database performance analysis | To optimize database calls |
| Multiuser database performance analysis | To optimize parallel processes |
| Single-user ABAP performance analysis | To analyze the use of ABAP language elements |
| Multiuser ABAP performance analysis | To look at SAP locks and memory consumption |

calculate the resource requirements for a multiuser environment.

In conducting the single-user tests and performance measurements, you generally want to think small and forecast high. This means that from your single-user findings, you want to be able to predict how the system is going to behave later on — i.e., you'll use the resource consumption of specific transactions to predict the resource consumption when these transactions are used by many users.

### *Conducting the Single-User Performance Analysis*

The single-user mode strategy for performance analysis is somewhat bottom-up. This is because when you interpret your single-user findings, you do so with a view to multiuser performance — i.e., you look at the database before you look at the ABAP code, as shown in **Figure 8**. For this reason, any database tunings you perform in the single-user mode can have an influence on all users, and are therefore important for your actual multiuser production system. The performance analysis that is done in this step is a large topic, which is beyond the scope of this article — we will discuss it in detail in a future article.

### *Measuring the Performance in Single-User Mode*

When you are satisfied with the single-user performance of a transaction or business process that you've tested on the development system, you should measure (also in single-user mode) the CPU, database calls, main memory consumption on the application servers, and the expected database size. By studying the results of these measurements, you may be able to identify possible weak spots in your transaction design.

For example, suppose a transaction that was designed for 1,000 users happens to block a central resource for 3 seconds. This means that when those 1,000 users run the transaction simultaneously, the central resource will probably be blocked for about 3,000 seconds per hour, causing long waiting lines.

In this case, you have two choices: either you optimize the code, or you go back and rethink the business process you designed. The same approach applies to memory or CPU consumption as well.

### *Using the Single-User Test Findings to Make Predictions About the Load Tests*

You can use your single-user test findings to predict the performance of the large load test system during the multiuser load tests. Here are some typical examples of forecasts you might make:

✔ **Memory consumption.** If one batch job fills one internal table with more than 500 MB in multiuser mode, you can predict that you will not be able to run more than two or three jobs simultaneously on a single application server due to the limitations of a 32-bit operating system.

✔ **Average CPU consumption.** If the results of your single-user measurements show that one sales document consumes 1 second of CPU, you can predict that 100 sales orders will probably consume about 100 seconds.

✔ **Maximum CPU consumption.** Suppose you have written an application that has to process 1 million line items in 5 hours. Your small-scale measurements indicate that the CPU needs 1 second to process 1 line item. This means that it would take 250 hours to process 1 million line items. For parallel processing, therefore, you would need more than 50 CPUs.

You then compare the results of the single-user findings with your predicted figures, and when you're satisfied you can proceed to tuning your load test system.[11]

### *Tuning the Load Test System*

Once you have tested, analyzed, and optimized your business processes in single-user mode, you'll want to make sure your load test system is ready for the multiuser load tests. If you held off creating the system (in step 8), set it up now.

Tuning a system for heavy multiuser load requires extensive experience in laying out the disks and network (e.g., load distribution and bandwidth), and also in tuning the operating system, the database, and the SAP system. For this reason, you may wish to consider using a two-team approach to preparing for the multiuser load tests. The best approach is to form two small special teams by carefully selecting a handful of application and performance specialists from your Application and IT project teams. You can then task the special teams with using the single-user test findings to set up the multiuser load test environment (see **Figure 9**):

1. **Team 1**, consisting of an *application developer* and a *performance specialist*, participates in the single-user test (step 11), helping to:
   - Conduct the single-user performance analysis
   - Conduct the single-user performance measurements
   - Make the predictions for the load tests
2. **Team 2**, consisting of an *application developer*, a *performance specialist*, and a *system administrator*,[12] works closely with Team 1 in the single-user test time frame (step 11) to:
   - Make sure master data and customizing settings have been imported correctly
   - Install and configure the load test system, based on Team 1's performance measurements and predictions
   - Set up the test environment for the multiuser load tests
3. **Team 1** and **Team 2** join forces to conduct the multiuser load tests (step 12).
4. Finally, **Team 1** and **Team 2** evaluate the results of the multiuser load tests (step 13).

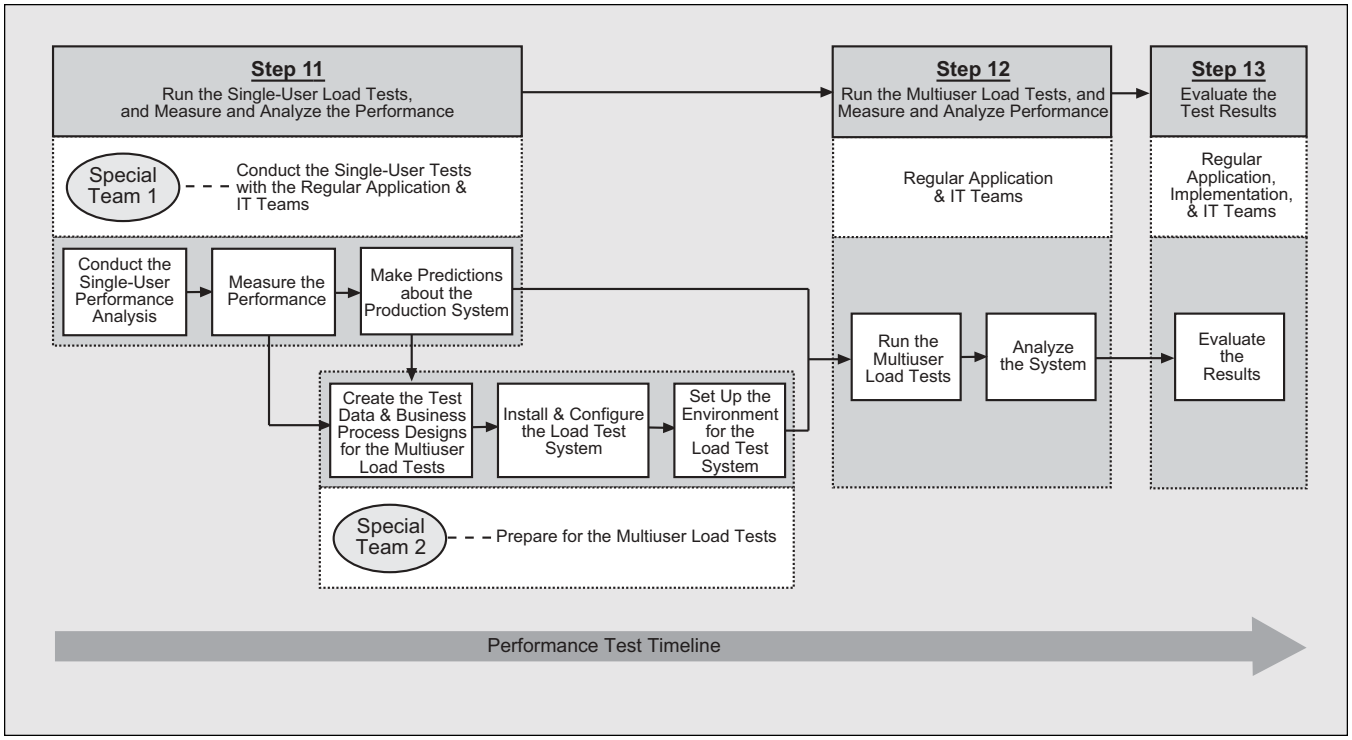
Note that these special teams will be active during the single-user test time frame: Team 1 will work with the regular Application and IT project teams to conduct the single-user tests on the development system; and Team 2 will get to work *after* the single-user performance measurements have been taken. Team 2 will use the measurements provided by Team 1 to create the data and business process designs for the multiuser load tests on the load test system.

[11] Please note that a single-user test is part of the GoingLive Check™.

[12] Your hardware partner can be involved in this step, too.

*Figure 9 Using Two Special Teams to Set Up the Multiuser Load Test Environment*




## *Step 12: Run the Multiuser Load Tests, and Measure and Analyze the Performance*

We recommend running the multiuser load tests in phases, doubling the number of users in each phase, and making sure that the throughput increases by the same factor. By continuing to double the users and checking the throughput until the CPU is fully utilized, you will gather test results that can provide you with valuable information about the scalability of your SAP system — e.g., you'll learn whether parallel processes are possible, and how your hardware behaves under the load.

Remember, this is an iterative process. In the first phase, you start by testing a given process using different sets of data, and you measure the response times and system throughput. Then, in each subsequent phase, you double the load.

If everything works all right, the throughput should increase with the same factor as the load until the database server CPU is fully utilized. If you do not see this scalability, it could be that you have an artificial locking situation due to the data basis you are working on. Otherwise, it could be that a bottleneck is being generated by some component of your system, such as the network, disk I/O, or CPU.

*We recommend running the multiuser load tests in phases, doubling the number of users in each phase, and making sure that the throughput increases by the same factor. By continuing this way until the CPU is fully utilized, you will gather test results that can provide you with valuable information about the scalability of your SAP system.*

### *Helpful Hints*

✔ Warm up the system. The load tests provide valuable knowledge about the configuration of your production system. Keep in mind that because of the SAP architecture, a lot of data that is shared by all users (e.g., screens, tables, and data descriptions), and all programs will be loaded on demand into various buffers on the application server. Therefore, your multiuser load tests will show poorer results if you conduct them on a "cold" system (i.e., one that has empty buffers) as opposed to a "hot" system (i.e., one that has been warmed up with test runs and is already in a steady state).

✔ When the results of these tests require you to perform an optimization, we highly recommend that you document any changes and tuning efforts, and their effects on the system behavior — this will enable you to more easily analyze what went wrong and why, and you'll find the documented information very useful when you prepare your system to go live.

*When the results of these tests require you to perform an optimization, we highly recommend that you document any changes and tuning efforts, and their effects on the system behavior — this will enable you to more easily analyze what went wrong and why, and you'll find the documented information very useful when you prepare your system to go live.*

## *Step 13: Evaluate the Test Results*

Finally, all three teams get together to review the results of the multiuser load tests to see whether they are ready to go live with the SAP system.

Your test was successful if the results (throughput, response time, etc.) are in line with your documented predictions and expectations. If the expectations set in the beginning have been met, the test was successful. Otherwise, analyses should be made as to why the goals of the load test were not met — go back through the 13 steps and rethink each step.

For example, perhaps there wasn't sufficient hardware, or it wasn't laid out properly. Maybe the business process tested contained errors. Another problem might have been caused by improper data constellation — perhaps locks were held too long or maybe the buffers were set incorrectly. Rethinking is the key here.

## *Helpful Hints*

✔ Starting with the project teams, make sure you don't underestimate the time needed to perform the load test. The very detailed project plan shown in Figure 4 projected 2.5 months. If at any stage something goes wrong, you can very easily end up with considerable delays. Therefore we recommend you conduct each step thoroughly, and make sure you define a sufficient number of milestones during the preparations. Often, too, the load test expert comes onboard at a late phase of the project when severe mistakes have already been made, so make sure you have the right people participating from the start.

✔ Don't be too ambitious with the performance you plan to test. Always bear in mind that the load test serves as the basis for going live. So, the only optimizations that you should be applying are those that can and will be used in the production environment, too.

✔ When preparing for a load test, your project plan is key. By carefully setting up a detailed project plan, you can ensure that the huge effort you are investing will have meaningful results.

✔ Do not underestimate the effort involved in achieving a well-planned and well-conducted load test. You need to be a perfectionist and avoid falling prey to false conclusions. If you haven't mapped your business processes correctly in the initial phases of the project, for example, when you scale up your single-user measurements, your forecasts may turn out to be incorrect.

✔ While you need to test those business processes that are going to be used for the production system, at the same time, you should keep things simple. It is next to impossible to simulate all the business processes for an entire day, and you don't need to test everything — for example, while it's important to test a process like POS Inbound, you really don't need to test something like the calling of help screens.

✔ In the multiuser tests, be very careful not to create artificial bottlenecks — for example, by doing all the work on one cost center or one customer. Make sure that the data created during multiple tests has no impact on further testing — for example, if you create open items in FI but never balance them, they may become an impediment to reporting and processing open items.

Furthermore, it is very important to think about probabilities of certain attribute combinations, such as how many different customers will order how many different articles per day. This has a big impact on table sizes as well as response times for reporting on the SAP Business Information Warehouse.

✔ Be sure you clearly formulate and write down the objectives for your load test, and carefully document every tuning measure you make, every item of data you change, etc. The importance of this cannot be overemphasized. You can then use this information either to modify your expectations when you go live, or you can keep it as the basis for your production system.

✔ And last, but not least, a load test serves as a brilliant opportunity for you to train both your Application and IT teams for going live.

### *For More Information…*

For additional reading on performance and sizing, please refer to the following sites on SAPNet (to access these pages, you need an SAPNet user ID):

- **http://sapnet.sap.com/performance**
- **http://sapnet.sap.com/sizing**

If you are interested in further and more detailed information on how you can improve the performance of your SAP system, we suggest you refer to the online help for the respective transactions, and/or the SAP library.

## *Conclusion*

We hope this article has helped set down a process for successfully conducting a load test. We think the pragmatic approach we've illustrated will help you avoid many mistakes, of which the biggest and most frequent is that at least one of the steps described here hasn't been considered.

At the Performance & Benchmark Group, we have become involved in many load tests at a very late stage, and we have seen firsthand what happens, so the urgency with which we stress many of the steps here stems from experience. It often seems that load testing has become *l'art pour l'art* rather than a process focused on well-specified needs, and while for many large customers it is always a good idea to incorporate load testing into a project implementation, it should be thoroughly analyzed beforehand. You certainly do not need to test business processes that resemble standard application programs, for example, but if you have created your own coding, you should definitely analyze the processes in single-user mode.

We find that most customers underestimate the time and skill required to conduct a custom load test, and in such cases, vendor experts, SAP, or consultants are called in to help solve problems that wouldn't have arisen had thorough research been in place at the beginning.

If you've carefully considered the 13 steps we've outlined in this article, you should now be poised for a smooth rollout of your system. Good luck!

PS: Just to answer the article's initial question, yes, a performance test is for you. We strongly recommend a single-user performance analysis as conducted in SAP's GoingLive Check™ service. A multiuser load test, however, should be very carefully considered before you commit to the effort and expense that comes with it.

*Susanne Janssen has been with SAP since 1997. She co-developed the Quick Sizer tool and is responsible for information development in the areas of performance, standard application benchmarks, and sizing. She can be reached at susanne.janssen@sap.com.*

*Dr. Ulrich Marquard joined SAP in 1990. He is head of the Performance & Benchmark Group, and does extensive research in the fields of system architecture, system analysis, scalability, and performance. He can be reached at ulrich.marquard@sap.com.*