Test Plan

The test plan covers the test strategy for client applications, how the test environment is configured, and how the team will test the installed/migrated database and support scripts. It describes the tools, methodologies, best practices, events, resources, and schedule for the test effort.

Developing the Test Plan

The test plan should address the following areas. Many of these items are described in more detail later in this guide:

- Test strategy and scope
- Phases or categories of testing
- Testing resources
- Testing schedules that detail the sequence of tasks that the test team will follow to accomplish the testing
- Any dependencies that test cases have on each other and other resources (for example, technical documentation, facilities, and so on)

Creating Test Cases

Test cases should be based on usage scenarios. The test cases include:

- Setup instructions, if necessary
- Step-by-step instructions for executing the tests
- Expected results

Try to reuse test cases from existing system documentation. If there are no existing test cases, it may be necessary to develop new test cases. These could be based on the user requirements, user documentation, or, to some extent, on the user interface (UI).

Testing Categories

Testing and test cases can be categorized according to the aims of the tests being executed at a particular point in time. Many test cases will span the different types of testing being performed, whereas others may be more specifically targeted at one particular type of testing.

Unit Testing

Unit testing ensures that the individual components that comprise the solution work as expected. In a project, unit testing will encompass testing that stored procedures, triggers, and tables (and their data) have been transferred correctly, and work as they did before the installation/migration. If the client has also been migrated (instead of retargeted), developers will unit-test each component of the client as it is migrated; if it's a custom environment case.

System Testing

The purpose of system testing is to verify the end-to-end functionality of the system. This is sometimes referred to as Integration Testing. System testing is designed to test *all* of the system as a whole, and not just individual components. If possible, try to perform testing in parallel with development so that system testing does not unnecessarily delay the project.

System testing will also ensure that the migrated application works with all of the external components. A warehouse management system, for example, may have to integrate with programmable logic controllers that handle conveyors. Sometimes these external components will need to be simulated. However, there is also a requirement that you have run the system end-to-end with actual instances of the external hardware. In circumstances where this hardware is not shareable with production, you will need to schedule time on that hardware with the production department. Because such system tests may be extremely expensive, it is important to ensure you get as much out of such tests as possible. A test that is supposed to last for several hours and fails fatally five minutes into the test has wasted time and money.

Stress Testing

It is important to properly test the system under realistic load before releasing to production. Realistic means at least 10 percent greater load than the currently anticipated peak load. You can use stress testing to troubleshoot database concurrency issues that may occur when there is a high level of concurrent activity. Issues with the database design may only become visible when you carry out testing under load.

One of the common problems uncovered by migration projects is in relation to concurrency and the fact that the new hardware and software platform behaves differently under high load. These problems may have always existed in the application, but have never been seen before because the old hardware was slower, or the system used a different scheduling algorithm, and so on. For this reason, you should begin stress testing as early as possible, as a part of the system test, in parallel with the development effort.

Utility Testing

You should test important tools such as schedulers and backup software based on production usage scenarios.

Security Testing

Testing for deployment-related security issues should ensure that sensitive data is only available to verifiably-approved users. In migrating from Sybase to SQL Server, you will have changed the location of the database. The network path must be secure to prevent snooping, and the security domain that is hosting the AD and SQL Server database must correctly authenticate and authorize user requests.

Inside the database, users and groups will have been mapped to SQL Server users and roles. Testing must ascertain that the mappings have been performed correctly and that all users and roles have been granted only the appropriate privileges, no more and no less. Furthermore, permissions granted over objects in the SQL Server database must be thoroughly tested.

Security requires both that authorized users be able to access certain features, and that unauthorized users are not able to access those features. Security testing tends to concentrate on the second of these requirements, but the first should not be forgotten. A system that defends itself by turning off features while under attack is vulnerable to a denial of service attack (DOS Attack).

High Availability Testing

This testing ensures that disaster recovery plans are functional and work as intended. For systems using clustering, you will test the failover functionality. For systems using log shipping, you will test recovery and reconnection to the log-shipped server. Other tests will need to be conducted if storage area network (SAN)-based clusters or a third-party solution is used.

Aspects of high availability testing are required even for applications that are not required to be highly available: do the backup and restore procedures work correctly? How long does it take to restore data in the event of a disaster?

User Acceptance Testing

User acceptance testing validates that the business requirements have been met. It also gives users and support personnel the opportunity to understand and practice the new technology through hands-on training.

User acceptance testing can use test cases that are similar to the tests performed during system testing; however, it is important that the end user see exactly the same environment that he or she would see in the production environment.

Remember: the migration project itself should have added no new functionality to the system; if new functionality is required, it should be a post-migration project. Therefore, user acceptance testing will likely focus on user interface, performance, and security aspects of the new system.

Regression Testing

In the event of the system being updated or corrected after fixing bugs, a suite of test cases must be available that ensures that the system still operates as required. You execute these test cases as a regression test. The results should be logged and verified against expected or previous results. Any differences in output must be accounted for.

Just because a test passed on a previous release of the migrated application is no guarantee that it will pass on a later release. Regression tests should be automated to the maximum extent possible and run many times.

Because the migration project should have added no new functionality to the application, regression tests can often be run against the original system, as well. This provides a good baseline for acceptable test results. Please also be aware of invalid tests.

Deployment Testing

You need to test the scripts, methods, and processes associated with the deployment itself, and then test any potential fallback eventuality. These tests are typically conducted on the stage environment.

Any data replication between the AD and SQL Server systems should be tested.

Note Deployment scripts should always run free of any error messages, no matter how trivial. For example, if a script drops a table that does not exist before recreating it, a developer may consider that as trivial. Such messages will hide any real problem, and deployment testing should flag such cases as errors in the script.

Resources

Performing adequate testing requires that you plan for the availability of the necessary equipment, time, and personnel.

Testing Team

You must carefully select the testing team. An inexperienced testing team may not be able to test the system thoroughly.

A tester should be familiar with the technology, the business, and the customer requirements. Testing a Microsoft SQL Server database requires that the tester be proficient with the architecture of SQL Server and the integration with Microsoft Windows Active Directory Domain Services as well as Federated Services.

A developer should never be a part of the testing team (beyond developing and running the unit tests). Developers know too much about what is happening internally in the system and will inherently avoid doing certain things that are "wrong." Testers have two goals: to ensure that the system *does* what it is supposed to do, and to ensure that the system does *not* do anything it is not supposed to do.

Deployment Plan

This section provides information to help you plan your deployment strategy for the databases you are migrating or a new AD you are placing in. How that takes place, the order of events, and when the deployment is judged successful will be different in every situation.

The deployment plan documents the strategy that you decide upon and that you will use to prepare end users and operations personnel before and during deployment. Its creation is driven by the Release Management Role. It details all the steps of the deployment process and contains information about the planning, organization, and realization of the following components of the migration:

- **Site survey**. What hardware, software, and management infrastructure is currently in use and how should the new system fit in with this configuration?
- **Hardware and software installation strategies**. How will the required hardware and software needed to support the new system be rolled out?
- **Deployment mechanisms**. How will the system itself be deployed across the organization?
- **Resources**. Which personnel, time, and other resources will be required and when?
- **Contingency plans**. What happens if the deployment does not proceed as planned?
- **Systems support**. How are user and system issues handled, and what service-level agreements are in place?

  Note The deployment plan is a living document that should be continually updated to reflect changes that are made if testing of release-level components reveals a deficiency.

The planning activities that result in this document include a number of steps, as follows:

- Define the roles and responsibilities of your deployment team and the subsequent operations team. Decide upon the membership of this team, which may or may not overlap with your development teams.
- Define supplementary requirements, such as training, support, communication requirements, and supporting documentation.
- Consider what fallback strategies you are planning.
- Define the deployment steps; for example, whether to conduct a staged cutover, which may incorporate elements of the pilot application.
- Articulate these goals in the form of a deployment plan. The deployment plan should include a document that describes all of the steps of the deployment process.

- Schedule the deployment and allocate appropriate resources.

Surveying the Target Environment

Because it is likely that the operation of production systems is owned by a group separate from the development group, a formal approach must be used during the transition from development to production. The approach usually involves communications, planning, agreement on both terminology and methodology, and prescription of test sequences and expected results.

It is possible that the target environment has been built within an existing data center consisting of other components and systems. The data center may maintain a heterogeneous environment because of some of the choices made in the migration. The impact of component and system relationships must be considered in the target environment and deployment testing. This is almost certainly one difference from the staging environment and may require consideration in test plans.

Parts of a successful deployment include properly migrated systems, properly created and specified acceptance tests, and a properly specified and built target environment. Performing a detailed survey has several advantages:

- By setting the scope of the tests to be performed while deploying in the target environment, you can determine what elements of the infrastructure are outside the scope and, therefore, do not need to be considered. This is particularly important in more complex environments in which it may not be absolutely clear where the boundaries of functionality or responsibility lie. By setting the scope of the tests to be performed in the target environment and getting operational managers and end user representatives to agree with the extent of the scope of testing, you can avoid problems later.
- When you know exactly what elements lie within the scope of the deployment testing, you can determine what impact the deployment may have on any of these elements. You can ignore those elements that lie outside the scope of the specified deployment tests.
- It is also possible to determine which elements of the existing (pre-migration) environment will have an impact on the migration. For example, the existing environment might impose constraints in terms of network bandwidth, storage, or processing. Remember that if you choose to conduct a pilot, or you run the migrated application in parallel with the existing application, you will need to allocate additional infrastructure resources to accommodate the two platforms operating simultaneously.

Some information for the detailed survey should already be available as output from the environmental assessment you performed earlier. However, depending upon the amount of time elapsed between the environmental assessment and the time of this survey, the assessment information will need to be updated to ensure that it is current, to gain a final confirmation of the details, and to take into account any changes that have been made in the course of the migration project.

The following items should be considered within the scope of the survey:

- **Hardware**. In particular, the servers that are used for hosting the AD or databases to be migrated should be considered. Also include storage hardware, which may be connected directly to the database servers or consolidated as a storage area network.
- **Network**. This includes such low-level networking devices as switches and routers, and such high-level devices as firewalls and proxy servers.

- **Software**. This includes such low-level software components as application servers, Web servers, and operating systems, and such high-level components as off-the-shelf and in-house software packages.
- **People**. This includes the users of the applications to be migrated and any application that might be affected by the migration and operational and support staff who may be involved in the migration or the support of the resulting application.
- **Physical infrastructure**. This includes buildings, rooms, and fittings, as well as the geographical locations of all infrastructure elements.

Record the survey results and review them with management and user representatives to ensure the data captures all items relevant to the migration.

Following this review, each item can be assessed to determine what the impact of the deployment would be. Impact should be considered in terms of functionality and also in terms of delivery-of-service levels, which include performance, scalability, security, manageability, and usability. The impact assessment should be conducted involving those roles responsible for the specific elements.

The impact assessment will enable you to determine any new requirements for hardware, software, or physical infrastructure. It will also enable you to sequence the different elements of the migration. Placing the elements of the migration in a particular order is generally driven by one of the following:

- **Deployment order**. For example, you may need to deploy the hardware before you can deploy the software.
- **Deployment priority**. For example, certain databases are more important than others, and will, therefore, require migration first.

The impact assessment should also enable you to determine the risks to the migration itself. You will need to mitigate these risks as part of the migration process; for example, by defining an appropriate fallback strategy.

You should gather and analyze the following information:

- Database server and databases:
    - Number of servers to be migrated and consolidated
    - Number of databases to be migrated and consolidated
- User impact:
    - Database user accounts
    - Microsoft Windows domain user accounts
    - Windows domain user groups
    - User account security, audit policy
    - List of order for account transitions
- Application impact:
    - Number of applications
    - Importance of applications: that is, ranging from important to low-priority
    - Scope of application: organization-wide, department-wide
    - Required changes: data source definition, database connection string, COM+ packages
    - Estimated transition and downtime
- Database management and operations impact:
    - Operations management and deployment tools, for example, Microsoft Operations Manager (MOM) and Application Center
    - Database administration scripts, tools

    - Administration task such as backup and restore procedures
- Network impact:
    - Domain, trusted relationship
    - Firewall, proxy server
    - Network traffic, performance
- Geographic location changes
    - SIP Agents Deployment Assessment (Software)

The impact assessment should help you to identify additional requirements that will ease the migration and enable the resulting applications to run more smoothly. The additional requirements include:

- **Training**. This includes training on SQL Server, Windows, SCCM, Lync, SharePoint, and other support tools such as Application Center and MOM if appropriate.
- **Support**. Call your local Microsoft representative about customized support options.
- **Communications**. Procedures and infrastructure must be in place to allow all roles and users affected by the migration to be able to communicate effectively and in a timely manner.
- **Preparation**. Release preparation should include, for example, delivery and checking of hardware and license keys for software.

Validating the Target Environment

If the owner of the production environment rather than the development team for the migration project builds the new hardware in the target environment, the development team should perform a survey of the target environment after the new hardware has been installed and is considered operational, and before actual deployment testing commences.

A comparison of components installed versus specified component requirements should be verified. An operational test should be performed on any components or systems that are to be included in the deployment's acceptance test. This will ensure that acceptance tests will be performed on the specified target environment and ongoing operation of the environment conforms to the physical design that was part of your solution design.

Defining the Deployment Steps

The process of surveying the target environment should enable you to gain a full understanding of exactly what the migration is supposed to achieve, and it should reveal what kinds of support the migration requires to achieve its intended result. You can use the information that you gather as a result of the survey to define the steps that are required to enable the migration to take place.

You can choose from a number of different options for the deployment, both in terms of how to perform the cutover and how you plan to fallback in case you experience problems. To ensure that you achieve a successful migration, you need to choose deployment options that take into account the risks that you have identified.

Define the Cutover Strategy

There are a number of options to take into account when defining your cutover strategy. First, decide whether to replicate the database that you are migrating. Second, decide whether to phase in the migration or to complete the migration in one full step.

Straight Cutover

It is possible to migrate from one environment to another without taking any special steps. A *straight cutover* is an option in situations in which downtime is not an issue, such as during the development of a new site, when the databases concerned are not mission-critical, or when a suitable window of opportunity is available for the migration (for example, during a holiday period).

The disadvantage of this approach is that it is extremely high risk; If there is a delay in the middle of the cutover process, you might miss the window of opportunity. If, despite all the testing, the new application fails to operate correctly in production, it is much harder to revert or fall back to the original system. To minimize cutover risk, you could consider duplicating the hardware that is required to run the database server, but this might not be a viable option in all situations. In any case, you should keep a verified, last known good image of each database server so that it can be restored with minimal problems in case of fallback.

Parallel Cutover Using Replication

Using replication as part of a migration is an appropriate technique for organizations that cannot afford the downtime that is required to build and then migrate a database. By first replicating the original database, the database replica can then be modified without affecting the users of the existing database.

The disadvantages of performing a migration in parallel include that it:

- Requires that sufficient database server resources (that is, hardware platforms) are available to run two database installations in parallel.
- Requires that you ensure the availability of mechanisms to guarantee the equivalence of the two data sets.
- Requires that you configure replication software for Microsoft SQL Server.

Serial Cutover Using Phasing

Using a phased cutover is an appropriate technique in situations in which there are a number of databases to be migrated, where the users of each database are independent of each other, or when the time that is required to perform the migration is available in short discrete periods. In such situations, it is possible to define a subset of the databases that can be migrated in a short time frame. There may also be scope for deploying core elements of the migrated application framework; for example, the database servers could be deployed, tested, and even approved before the databases themselves are migrated.

It may not always be possible to phase the migration, either because of the complexity of the databases to be deployed, or their interdependencies with other infrastructure elements. In addition, the issues that were listed for the straight cutover are also true for a phased approach.

Review Fallback Strategies

The fallback plan should allow the system to be rolled back in a controlled manner that provides the least disruption possible. If necessary, you want to be able to perform a graceful retreat instead of a chaotic rout. There are often workarounds that you can implement before deciding that it is necessary to fall back to the implementation.

There are two different fallback conditions that you need to consider: fallback during the deployment, and fallback after the deployment. During the deployment, you may have to fall back if some environmental or

external event occurs. For example, a power outage during the window of opportunity for the deployment may rob you of critical time, and you decide you need to back out and restart the deployment at a later date.

A fallback after deployment occurs if various tests have missed some critical aspect, and the new system does not perform or behave as the old system did. A post-deployment fallback plan will need to include methods for taking whatever operational updates have occurred to the database back to the Sybase system.

Which fallback strategies you can choose from will depend on the strategy that you select for cutover. In all cases, it is important that the Sybase database installation be kept up to date with the latest transactions, or is at least frozen in a consistent state allowing users to continue to work.

You should make a list of the potential failures that can occur during the migration process and document actions to be performed in the event that things go awry. For example, consider what would happen if something went wrong in the middle of the script sequence, or if the database scripts worked but the application team encountered errors and had to remove the changes. Always plan for a minimum of two possibilities: complete removal of all changes (which could require rollback scripts, or a restore, if the application is not designed to support multiple versions running in the same database), or an alteration of your plan because of predictable or unforeseen circumstances. Think through as many scenarios as you can so you are sure you know what to do; make notes in your scripts so that someone else can make changes in case you are unexpectedly unavailable during the actual implementation.

The roles and responsibilities of everyone involved in the deployment need to be clearly understood. There will likely be a number of key points during the deployment when a go/no-go decision is made; for each of these points, who has input on the decision, and who has the ultimate authority?

Note Fallback strategies should not only consider the possibility of problems with the migration, but also the impact of external events. For example, there may be a change in the business that adversely impacts the cutover window, or there may be an unexpected number of necessary transactions as a result of an unexpected upsurge in sales.

Fallback after Straight Cutover

In the straight cutover scenario, the environment should be fixed at a moment in time before the cutover was initiated. It should, therefore, be straightforward to revert back to the environment at the point at which it was left. However, the longer that the migrated system ran in production, the greater the likelihood that the data in the system would be out of date.

Fallback may require you to remake broken connections, such as those that may occur between client computers and the database. Should a straight cutover be proposed, it is important to remember the following points. During the cutover you should:

- Confirm that the server cutover is successful before clients are cut over.
- Cut over and test a limited subset of clients before cutting over the rest of the clients.
- Allocate sufficient time to allow for a fallback with minimal user disruption (given that a straight cutover may rely on a narrow window of opportunity).

In situations in which the same hardware is used for the SQL Server installation, you may need to restore an image of the installation from backup. In this case, ensure that you have diagnosed the problem as completely as possible before erasing the SQL Server configuration.

In case problems occur during a straight cutover, establishing a final go/no-go decision point toward the end of the cutover window should ensure that time is available for fallback, if necessary.

Fallback after Parallel Cutover

You can implement replication from SQL Server to the original database. This strategy will ensure that the installation is kept up to date for the duration of the cutover period and alleviates the time criticality of the straight cutover.

Assuming the replication software works properly, the main technical areas to be addressed for fallback are on the client side. A fallback will inevitably cause some user disruption, so it will be worth running a limited set of clients to test the SQL Server installation before switching the majority of the clients.

Fallback after Serial Cutover

A phased cutover also allows you to fall back to the last known consistent state, and then iron out problems before trying the migration again. Each phase can be considered as a mini-cutover, and so it should follow the same criteria as the straight cutover, as detailed previously.

Scheduling the Deployment

Scheduling involves predicting how much time the total migration deployment strategy will take given the complexity of the solution and choosing the best time to execute the plan. When predicting the schedule for a migration deployment, the following things should be taking into consideration:

- Business system availability requirements, as expressed by the users.
- Maintenance schedules.
- System downtime (for example, does the organization operate 24 hours a day, seven days a week, or does it operate Monday through Friday with weekend downtime?).

For each database server, you need to estimate the time that is required to complete the installation and migration. The estimates will be calculated based on the number of databases, clients, and applications that are running on or dependent upon the target server, as determined during the scoping of the migration. In addition, you may need to factor in extra time for complex databases and applications. The objective is to create a deployment plan that begins with the simpler sites, servers, and databases, and then progresses to more complex ones. Build a deployment plan and schedule that is practical.

Defining the Schedule

To plan and schedule deployment activities, you should break down the large deployment effort into smaller tasks and releases. Develop detailed schedules based on the milestones of the deployment process. These tasks could include:

- Installation and configuration of new SQL Server
- Migration of user databases
- Setup of user accounts, both SQL Server and Windows
- Migration of user applications
- Acceptance testing
- Transition and hand over

After you have defined a schedule, you can allocate resources to it. You should:

- Update the project plan with detailed allocation of resources.
- Check and confirm availability of resources, identifying any possible overuse of resources.
- Establish a baseline and track progress on tasks and budget.

  Note It is important to allow for flexibility in these plans and schedules because business priorities and test results could impact the order and timing of migration.

Scheduling Considerations

This section covers the issues that should be considered when developing a schedule that addresses the client's business requirements for each migration deployment strategy.

Parallel Cutover Strategy

As discussed earlier in this chapter, the parallel migration approach is ideal for production servers that operate twenty four hours a day, seven days a week and for applications that require high availability. Scheduling this approach is very difficult because it requires the appropriate timing and monitoring before the execution of the plan. The following considerations need to be addressed and made a part of the execution task list when performing a parallel migration:

- Installing and configuring applications.
- Predicting the amount of time required to configure the application to work with the new environment.
- Reconfiguring heterogeneous replication of SQL Server. If the environment already utilizes replication, this effort will be a little easier.

  Note The process of developing and running the replication environment, performing validation and maintaining the required performance around the production environment, and creating fallback strategies requires significant effort.

Straight and Phased Cutover Strategies

As discussed earlier in this chapter, the cutover without replication and phased cutover migration approaches are ideal for production environments that allow for periods of downtime. Scheduling these approaches focuses on designating the appropriate "downtime" period during which to conduct the migration. The following considerations need to be addressed and made a part of the execution task list when performing the migration:

- The length of downtime that is available. (For example, will the system be down for an hour or over the weekend?)
- The amount of time that you predict will be necessary to conduct the migration, including all applications and interfaces.

  Note Be prepared to postpone or roll back at short notice, if necessary.

Transition to Operations

After the migration has taken place, the resulting AD, SQL Server environment will need to be handed over to operations staff. Allow time in the schedule for this handover, particularly if the operations team requires post-migration tests before accepting the delivery.

Reviewing the Plan

At the end of the planning process, review the plan to ensure that it meets the requirements of its users. This includes confirming:

- Migration roles and responsibilities.
- The scope of the migration, in terms of databases, software, hardware, and people.
- The strategy for moving from the other database environment to the SQL Server environment.
- The fallback strategy to be followed in case problems occur.
- The migration schedule and resources, in particular, ensure staff availability.

The plan should be reviewed by those individuals and teams that it implicates. In addition, the plan should be signed off by the main sponsor of the migration, who may be a senior user representative or an IT budget holder.

Cutover Strategies

A cutover strategy is the process of moving the migrated database to its new production environment. Three cutover strategies have been successfully used during migrations:

- Parallel cutover with replication
- Cutover without replication
- Phased cutover

Whichever cutover strategy you choose, you should include the following tasks in your plan:

- Pre-migration tasks
- Network communication requirements
- Preparation of the existing Sybase environment
- Configuration of the new SQL Server production environment
- User acceptance testing
- Database synchronization
- Migrated application configuration
- System monitoring

Fallback Strategies

When dealing with valuable data and systems that can only be down for a limited amount of time, as is the case during a database migration, it is imperative to have a solid fallback strategy in place and ready to implement. A fallback plan is designed to restore the other databases and all associated systems to a known state in the event that unanticipated problems occur during the cutover.

"Fallback Considerations," discusses factors that you should take into account and various strategies to protect yourself against a failed migration. This appendix includes information about the following topics that are relevant to a fallback plan:

- **Uptime and outages analysis**.
- **Database synchronization**.
- **Types of fallbacks**:
    - **Fallback for a parallel cutover migration**. Uses heterogeneous replication to maintain database synchronization.

    - **Fallback for a cutover migration**. Uses an effective backup and restore strategy to maintain database synchronization.
    - **Fallback for a phased cutover migration**. Uses an effective backup and restore strategy to maintain database synchronization.
- **Fallback scenarios**. Scenarios that initiate the fallback plan.
- **System monitoring**. Monitoring processes that ensure the stability of the migrated system and that trigger implementation of the fallback plan.
- **Protocols**. Escalation status of each problem identified during and after the migration.
- **Code freezes**. Defined configuration, or state, for the migration.
- **Execution plan**. Defined process for implementing a fallback plan.
- **Application shutdown and setup process**. Shutdown and reconfiguration process that restores applications to the original system.

Support Plan

The support plan describes how problems that arise with the migrated database will be resolved. The plan specifies a process for users to raise issues, and how these issues should be resolved.

The end-user support plan identifies the resources, processes, and skills that users will require to properly use the system. It describes the different elements of user support that will be provided in connection with the migration, which might include any of the following types of documentation:

- Reference manuals
- Help
- Online tutorials
- Training

There will be interdependencies between the support plan and the training plan.

Creating the Project Schedules

After you have completed the initial versions of the conceptual design and functional specification, you can map the individual functional components to specific tasks and assign the tasks to the teams that are responsible for developing those components. Each team lead prepares a plan or plans for the deliverables that pertain to their role and participates in team planning sessions. Examples of such plans include a deployment plan, a test plan, an operations plan, a security plan, and a training plan. As a group, the team reviews and identifies dependencies among the plans. The Master Project Schedule combines all the schedules from the various teams.

One of the job aids is a Microsoft Project file that shows an example master schedule for a Sybase database migration project. The job aids are available in the Job Aids folder in the download version of this guide.

Interim Milestone: Master Project Schedule Baselined

After the team approves the specifications, plans, and schedules, the documents form the project baseline.

Closing the Planning Phase

At this stage of the project, you have assessed which databases and client applications should be migrated, and you have decided an appropriate strategy for each one. You have reviewed the choice of client

connectivity technology to be used and ascertained the feasibility of the chosen solution. You have used this knowledge to create the Project Master Plan.

Key Milestone: Project Plans Approved

At this point, the project team and key project stakeholders agree that interim milestones have been met, that due dates are realistic, that project roles and responsibilities are well-defined, and that mechanisms are in place for addressing areas of project risk.