Data Warehouse Software Maintenance Strategy

BI Best Practice Team

Jason Gartner, Scott Maberry and Bill O'Connell

June 2005

June 2005

Table of Contents

Intended Audience

Primary

- Data Warehouse Program Managers
- Data Warehouse Technical Architects

Secondary

- Data Warehouse Database Administrators
- Data Warehouse Hardware/Software Support Specialists

Adopting a Software Maintenance Strategy for a DB2 UDB Data Warehouse

Overview

The purpose of this paper is to discuss software maintenance strategies for the data warehouse. Though composed of multiple technologies, the data warehouse will be referred to as a "system" maintained by skilled professionals. The role of the Data Warehouse Technical Architect will be discussed as the technical leader not only during design and implementation, but also during software maintenance so the technical wellbeing of the data warehouse is maintained. When applying software maintenance to the data warehouse, this paper will discuss optimal practices and encourage organizations to take an "adopter" strategy in the interest of maintaining service level agreements. The central principle of the strategy is to follow IBM's recommendations on specific combinations of database and operating system software release levels. Finally, organizations will be encouraged to build and maintain a software validation test suite to identify and resolve issues before they are introduced to the production system.

This paper is intended for Data Warehouse Program Managers and Data Warehouse Technical System Architects. Key roles such as Data Warehouse Database Administrators and Data Warehouse Hardware Support Specialists will also benefit from the discussion.

Organizational Readiness

Business users rely on the data warehouse as an essential tool needed to accomplish their daily tasks; therefore, a commitment to the availability and stability of the data warehouse is paramount. Often, the business and technical owners of the data warehouse will define a service level agreement that defines the expected levels of availability and performance. To deliver the agreed level of service, any change to the data warehouse must be carefully planned, performed and tested so that the expectations of business users are met.

Many IT organizations are structured such that the technical team that maintains the data warehouse is often a "virtual team" composed of personnel from several areas of the IT organization (e.g. UNIX Administration, Storage Administration, Database Administration, Applications Development). When it comes to design, implementation and ongoing changes to the data warehouse, all components that interact directly with the system must be considered. Given this, the role of a Data Warehouse Technical Architect is important for the overall technical wellbeing of the data warehouse. This role spans all functional areas (e.g. operating system, database, ETL, applications) and is critical in the process of understanding all system components, evaluating proposed changes and assessing the impact on the data warehouse. A Data Warehouse Technical Architect will have the leadership skills such that they can coordinate across organizational boundaries and will be empowered by management to do so. Where possible, a "dedicated team" reporting to a Data Warehouse Program Manager is ideal. This organization creates a team where all the required skills to maintain each component of the system is represented, has an in-depth knowledge and share a common technical leader – the Data Warehouse Technical Architect. The data warehouse is a system - have a Data Warehouse

Technical Architect lead the technical team during the deployment and maintenance of this critical business system.

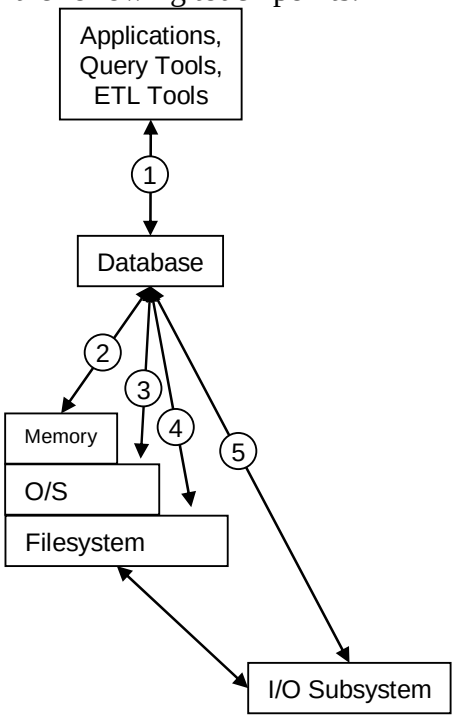
For more information on the topic of organizational readiness, ask your IBM Representative for a copy of the whitepaper titled, “Data Warehouse Organizational Structures” by Wendy Lucas of the IBM BI Best Practices Team.

The Data Warehouse System

The data warehouse is a system composed of hardware, software and various operating environments maintained by skilled professionals. When considering a change to the data warehouse, the impact of the change to the system must be assessed. For example, consider the following categories of software:

- Relational Database Management System and Tools
- Operating System and Tools
- Extract, Transfer and Load Tools
- Query Tools
- Applications

Since the data warehouse is a system, software and hardware components interact with each other. Consider the following touch points:



1. Applications, Query Tools and ETL Tools to Database
2. Database to Memory
3. Database to Operating System
4. Database to File System
5. Database to I/O Subsystem

Data Warehouses have many other touch points, but experience has shown that the touch points listed above are most important when making changes to software release levels. The various software components typically come from more than one software company; therefore, the Data Warehouse Technical Architect must assess the dependencies for each of the components and the impact of a planned change to the overall system.

The data warehouse is also composed of numerous hardware components. The major categories of components include servers, disk and tape storage and networking switches. Support for new hardware often requires upgrades to the operating system. Over time, firmware and/or device drivers are required to be updated. As the data warehouse grows, additional servers or disk capacity may need to be installed to support the system. As these changes are made, the Data Warehouse Technical Architect must assess the software dependencies and the impact of any planned changes on all the components.

As the data warehouse becomes more integrated into the fabric of a business, the availability requirements approach that of a mission critical system. To maintain the required availability of the system for business users, the process of installing, testing and promoting changes to the production system is critical. To facilitate this process, various operating environments are necessary including development, system test, and production. All of these environments should be included in the initial architecture of the data warehouse. Some organizations have found multiple test environments (i.e. Functional/Performance Test and Application Unit Test) to be useful to the data warehouse support team. Refer to Appendix C for a detailed discussion on validation testing and test environments.

Maintaining the Data Warehouse

Maintaining the data warehouse involves various changes. Take for example:

- Creation and alteration of data tables (i.e. their definition, layout and content)
- Amendment of access rules (i.e. authentication, authorization)
- Addition of items for performance reasons (e.g. indexes, materialized query tables, multidimensional clustering)
- Changes to database configuration parameters and parameters for the applications and utilities
- Changes to software release levels

This document concentrates on changes to software release levels that could be needed for a variety of reasons, such as:

- Adding new applications that have specific software prerequisites
- Supporting new releases of a software product that adds features or functions
- Resolving a problem that is discovered
- Applying maintenance to stay within recommended or supported levels of software
- Supporting new hardware devices

Regardless of the reason for making a software change, the decisions of how and when to make these changes must be determined by each organization. For the purpose of this paper, it is useful to characterize the software maintenance strategies as follows:

- Early Adopters
- Adopters
- Late Adopters

The "Early Adopter" is the organization that is willing to make software changes as they become available because they need/want the latest software functionality and maintenance. The motto of the Early Adopter is "when can I get it?" In this case, the organization either desires the latest in functionality or has selected a proactive strategy to apply software maintenance to avoid known issues before they encounter them. While having the latest functions and fixes, this strategy is not without risk since stability can be impacted by unsuspectingly introducing new problems.

The "Adopter" is the organization that makes software changes when there is a reason to believe that the changes will not introduce issues or impact stability. The motto of the Adopter is "let's wait and see." With this strategy, the organization benefits from the experience of other organizations that have already made similar changes and achieved stability. This strategy postpones the benefits associated with new software functions that are included in new software releases. While not that far behind, the adopter waits to install software maintenance until recommended.

Finally, the "Late Adopter" is the organization that only makes software changes when absolutely necessary and adheres to the motto of, "if isn't broken, don't fix it!" Making infrequent changes certainly reduces the chance of introducing issues or instability; however, sometimes the Late Adopter finds that they are out of support or have missed opportunities associated with new software release levels (e.g. performance improvements). When problems do occur or new functions are needed, this strategy results in significant changes being required, often associated with a business critical need.

	EARLY ADOPTER	**ADOPTER**	**LATE ADOPTER**
Description	Installs the latest software release levels when available	Installs the IBM recommended software release levels	Only installs new software release levels when absolutely necessary
Motto	"When can I get it"	"Let's wait and see"	"If it isn't broken, don't fix it"
Pros	Has the latest in software function and fixes	Benefits from additional testing by IBM and the experiences of other customers	Limits the number of changes
Cons	May unsuspectingly introduce issues	Must exercise patience to gain access to new features and functions	Upgrades become more significant and possibly at unplanned times

An organization may select a strategy based on the operational environment. For example, an "Early Adopter" strategy may be appropriate for a Functional/Performance Test environment that is primarily used by the database administrator and system administrator to perform initial system testing of new releases of database or operating system software. An "Adopter" strategy may be more appropriate for the production data warehouse. Refer to Appendix C for a detailed discussion of operational environments.

Software Maintenance for IBM Products

As discussed, the data warehouse is a system and anything that interacts with it must be considered when making changes. For IBM software, DB2 UDB and the AIX operating system are critical software components to consider. To assist organizations, IBM recommends specific software release levels for DB2 UDB and AIX. The purpose of making these recommendations is to support the shared goal of data warehouse stability. IBM has an internal test suite of data warehouse workloads where combinations of DB2 UDB and AIX are rigorously tested. Recommendations concerning software release levels are made based on IBM internal testing and the experiences of data warehousing organizations that are "Early Adopters." The recommended software release levels are reviewed monthly and generally revised every three to six months. As such, an organization seeking to be an "adopter" would generally apply software maintenance to DB2 UDB and/or AIX no more than every six months, but no less than once per year.

The Recommended Base Software Stack for Business Intelligence (BI) Warehouses on AIX can be found at: http://www-1.ibm.com/support/docview.wss?rs=71&uid=swg21179515 Additionally, a list of known issues for DB2 UDB on various releases of the AIX operating system can be found at: http://www-1.ibm.com/support/docview.wss?rs=0&uid=swg21165448

For Linux, the Recommended Base Software Stack for Business Intelligence (BI) Warehouses can be found at: http://www-1.ibm.com/support/docview.wss?rs=0&uid=swg21192752

As recommendations change, e-mail notification is available by registering for IBM's "My Support" at https://www.ibm.com/support/mysupport/us/en. To receive e-mail updates, add DB2 to the profile (Selection Path: Software: Data & Information Management: Databases: DB2 Universal Database for Linux, UNIX and Windows), subscribe to e-mail and select the "flashes" category.

DB2 UDB software release levels are identified by version number, release number and fix pack level. It is useful to note that certain combinations DB2 UDB Version, Release, Fix Pack levels are functionally equivalent. For example, a new DB2 release can be equivalent to the previous release at the current fix pack level. A new DB2 UDB fix pack level is generally available every three months. The AIX operating system is identified by version, release and maintenance level (ML). A new AIX maintenance level is generally available every six months.

Both DB2 UDB and AIX have the concept of APARs (Authorized Problem Analysis Report). An APAR is a recognized problem that requires a specific fix. With DB2 UDB, a group of APARs are included in a fix pack level. Alternatively, an individual DB2 APAR can be requested from IBM through a special build process when circumstances dictate that it is not feasible to wait until the availability of the next fix pack or to install the latest fix pack. The special build will include the specific fix to the problem. A special build must be used carefully since only limited testing is performed by IBM prior to making it available. Special builds can be delivered as a fix pack image or as individual files. It is recommended that the fix pack image is requested. A fix pack image requires more time to prepare for delivery, but software management is improved. Late Adopters Beware: applying individual fixes through the special build process may not be feasible if the installed fix pack level is more than two to three levels behind what is currently available. Support for a special build ends one month after the fix for the APAR has been released in an official fix pack.

Planning and applying software maintenance to the Data Warehouse

At the conclusion of applying software maintenance, the goal is for the system to maintain its functionality, stability and performance characteristics. As much as possible, apply software maintenance as a planned project using recommended software release levels. The planning process is led by the Data Warehouse Technical Architect, but includes the participation of the entire technical team (e.g. Data Warehouse DBA, Database Hardware/Software Support Specialist). The results of the planning process should be a detailed project plan that includes tasks associated with preparation, installation and validation testing. Refer to Appendix B for an overview of software maintenance tasks.

During the planning process and ultimately the installation, consider these guiding principles:

- Adhere to change management policies and procedures. Standard practices such as making one change at a time certainly apply. Realizing that a change to one software product can impact another even though the effected product did not change leads one to test thoroughly after any change.
- When a decision is made to install new software release levels, continue moving forward by identifying and addressing any functionality, stability, or performance issues that may arise. In the case of an issue that cannot be resolved in a predetermined timeframe, have a plan to back-out all changes and return the system to its original state. Installation of prerequisites for a software product may influence the sequence in which changes are made.
- Install the recommended AIX maintenance level first when making multiple software changes. Following a period of time with no issues, install the recommended DB2 UDB fix pack level. AIX maintenance should be installed in an “applied” mode and only “committed” following a determination that the software maintenance will not have to be backed-out.

- Plan to keep the software release levels for the data warehouse "current" using the software release levels recommended by IBM for data warehousing. Recommended software release levels will generally be within one to two levels of the latest available level.
- Avoid installing software release levels that are more current than shown on the IBM recommended base software stack website. This will reduce the chance of unsuspectingly introducing issues.
- Avoid the temptation to install a more current software release level than recommended with the purpose of avoiding an upgrade sometime in the future. Install what has been tested and known to work together.
- Understand your risk. Several factors influence the degree of risk associated with software maintenance including the amount of time to plan, number of dependencies on other products and amount of time available for testing. Low risk changes have few, if any, dependencies and require less time to plan and test. Higher risk changes, such as major releases changes require more time to plan and test than lesser DB2 fix pack or AIX Maintenance Level changes.
    - Version and release changes (DB2 and AIX) – Version and full release changes have the highest degree of risk and the most dependencies with other products. In general, wait until the first DB2 fix pack or AIX maintenance level after general availability of a new version or release.
    - Fix Pack Changes – Risk depends upon the new function included in the fix pack. Unless the new function is absolutely required, follow the "recommended levels" to avoid fix packs that have major functional enhancements.
    - APARs, special builds – Individual fixes have a lower risk because only limited software code is changed, but note that less testing is done by IBM than for full fix packs, releases or versions.

Once planned and installed, the testing of new software maintenance is critical. The goal of software testing is to identify any problems with functionality, issues with stability or degradation of performance. While IBM tests the recommended software release levels for DB2 UDB and AIX, this should not be considered a substitute for thorough testing by each organization. Through experience, organizations know their workload and the conditions that make the system most susceptible to issues. Organizations are encouraged to develop a validation test suite to exercise these functions in test environments. Consider beginning, or enhancing an existing test suite, for use with the next installation of software maintenance. Over time, a robust validation test suite will evolve that will save testing effort and increase the confidence that applying software maintenance will meet the goal of functionality, stability and performance. Refer to Appendix C for more details on building a validation test suite.

Summary

The availability of the data warehouse for business users is of the utmost importance. To achieve this goal, view the data warehouse as a system and assign a Data Warehouse Technical Architect to oversee its technical wellbeing. When making software changes to the data warehouse, become an “Adopter” by following the recommendations for IBM products. Avoid being a “Late Adopter” by including software maintenance in the list of planned projects for the data warehouse. Finally, build, maintain and enhance a software validation test suite to identify and resolve issues before they are introduced to the production system.

Acknowledgements

Authors

Jason Gartner, IBM Manager, DB2 UDB System Verification Test
Scott Maberry, IBM Business Intelligence Architect
Bill O'Connell, IBM Distinguished Engineer, Business Intelligence Chief Technology Officer

Special Contributions

Chris Klein, Experian DB2 Database Administrator
Appendix on Software Maintenance Tasks

Wes Rhodes, IBM Managing Consultant and Executive Project Manager, Business Consulting Services
Appendix on Software Validation Test Suites

Peer Reviews

Chub Varga, IBM Business Intelligence Architect
David Bryant, IBM Business Intelligence Technical Advocate
Hok Chou, IBM Business Intelligence Solutions Architect
Ken Jesberger, IBM Business Consulting Services
Pavel Sustr, IBM DB2 Advanced Technical Expert
Renson Clouden, IBM DB2 Lab Services
Rich Koch, IBM DB2 Technical Specialist

Appendix A – Software Maintenance Scenarios

The following scenarios represent common situations where various software release levels and combinations of IBM software are being used. In response to these scenarios, the "recommend software release levels" will be referenced. These recommended software release levels can be found at the following IBM websites:

- Recommended Base Software Stack for Business Intelligence (BI) Warehouses on AIX http://www-1.ibm.com/support/docview.wss?rs=71&uid=swg21179515
- Recommended Base Software Stack for Business Intelligence (BI) Warehouses on Linux http://www-1.ibm.com/support/docview.wss?rs=0&uid=swg21192752

Scenario 1

- **Maintenance Strategy:** Adopter
- **Situation:** You are using the recommended software release levels and a problem is discovered
- **Description:** DB2 is at the recommended fix pack level on the recommended AIX maintenance level. A problem is discovered that is resolved with a specific fix to DB2 or AIX. The fix is included in a more recent DB2 fix pack or AIX maintenance level.
- **Recommendation:** In general, maintain DB2 and AIX at the recommended levels and only apply the specific fix. For DB2, discuss the possibility of a special build with your IBM Support Analyst. The special build would be installed on top of the installed recommended fix pack level and would only include the requested fixes. Carefully consider the use of special builds as there is limited testing performed by IBM. Consider how important the fix is to the system and whether there is a workaround. If there is an applicable workaround, use the workaround until the fix is included in a fix pack that is recommended. Discuss the relative risk that a specific fix may pose with your IBM Support Analyst. Not all fixes have the same amount of risk. For AIX, request the specific PTF (program temporary fix) and apply it to the installed recommended AIX maintenance level. In both cases, a more recent DB2 fix pack or AIX maintenance level might be available, but should only be installed after they become part of the recommended software release levels.

Scenario 2

- **Maintenance Strategy:** Late Adopter
- **Situation:** You are installed below the recommended software release levels and a problem is discovered
- **Description:** DB2 is below the recommend fix pack level on a maintenance level of AIX below what is recommended. A specific fix to DB2 is required that is included in both the recommended and latest DB2 fix pack.
- **Recommendation:** In general, upgrade AIX to the recommended maintenance level first. After operating without incident, upgrade DB2 to the recommended fix pack level. Avoid upgrading to the latest AIX maintenance level or DB2 fix pack level until they become part of the recommended software release levels.

Scenario 3

- **Maintenance Strategy:** Early Adopter
- **Situation:** You are installed above recommended software release levels and a problem is discovered
- **Description:** DB2 is installed above the recommended fix pack level on a maintenance level of AIX at or above the recommended level. The fix to DB2 is included in a newly announced fix pack.
- **Recommendation:** When operating above the recommended software release levels, each situation is different and needs be assessed based on the unique circumstances. In general, however, maintain the installed DB2 fix pack level and only apply the specific fix to DB2 by discussing a special build with your IBM Support Analyst. A special build would be installed on top of the installed fix pack level. The special build would only include requested fixes. If, however, the latest fix pack level includes APARs only, a different recommendation might be made after assessing the situation.

Scenario 4

- **Maintenance Strategy:** Adopter to Early Adopter
- **Situation:** You are installed at or below recommended software release levels and newly announced features are required
- **Description:** DB2 is at the recommended fix pack level on the recommended AIX maintenance level. Newly announced DB2 features are absolutely required by the project and are available in the newly announced DB2 fix pack.
- **Recommendation:** When considering software release levels above what is the recommended, validate that the new features are truly an absolute requirement and cannot be delayed until the fix pack becomes part of the recommended software release levels. If it is determined that the new features are required, install the newly announced fix pack on the currently recommended AIX maintenance level or the AIX maintenance level specified as the minimum prerequisite. When applying software more current than recommended, consider having a Functional/Performance Test environment where DBAs and System Administrators are the primary users. Be sure to include extra time in the project plan to perform additional testing. In the event of issues, plan to move forward by resolving any problems that may arise by applying special builds until the environment provides the required function, stability and performance. As always, have a back-out plan to return the system to its original state in case a problem cannot be resolved within a predetermined timeframe.

Appendix B – Overview of Software Maintenance Tasks

The following is an overview of common tasks to consider when developing a project plan for applying software maintenance to a DB2 UDB data warehouse running on the AIX operating system:

Planning Tasks

1. Validate the current implementation adheres to best practices
2. Determine if any contract resources will be required to complete the project.
3. Review and document all installed software and their current levels.
4. Review the IBM website for DB2 UDB for AIX operating system prerequisites.
    - http://www-1.ibm.com/support/docview.wss?rs=71&uid=swg21181544
5. Review the IBM website for recommended levels of DB2 UDB and AIX.
    - http://www-1.ibm.com/support/docview.wss?rs=71&uid=swg21179515
6. Review DB2 fix pack *readme* file for installation instructions, known problems and workarounds, and reference information.
    - http://www-306.ibm.com/software/data/db2/udb/support/
7. Contact your other software vendors, as appropriate, to determine support and stability for their products with the planned software release levels. While all software that interacts with the data warehouse is important, ETL tools require special attention since they are more tightly integrated with the database than with other products.
8. Contact the IBM Support Center if you are running a special build to determine if an updated version of the special fixes is needed. This is required so no special fixes are lost.
9. Develop a detailed project plan that identifies what tasks must be completed, when they must be completed and who is responsible for performing them.
10. Define a plan for how and when to back-out software changes should an irresolvable problem occur.
11. Develop a test plan and test cases that will exercise the changes and will help to identify unexpected changes in system functionality, stability or performance characteristics.

Preparation Tasks

1. Database Administrator - Run performance related baseline test cases before applying any software changes so a before and after comparison is available. (Note: Understanding the initial configuration settings and a log of any subsequent changes is important).
2. AIX Administration - Download fix pack image from IBM
3. AIX Administration – Upload and stage fix pack image to server(s)
4. AIX Administration - Prepare fix pack image (extract installation files from tar image)
5. Database Administrator – Backup the database

High-Level Installation Tasks

The following are typical tasks, but they can vary with each fix pack or special build:

1. AIX Administrator and Database Administrator – Review and follow all tasks and considerations documented in the fix pack *readme* file
2. AIX Administrator – Commit current AIX levels (this will allow a back-out of the new AIX maintenance level if problems occur)
3. AIX Administrator - Apply AIX prerequisites, as required (apply only; not commit)
4. AIX Administrator - Commit current DB2 fix pack (this will allow a back-out of the new DB2 fix pack level if problems occur)
5. AIX Administrator and Database Administrator – Stop all DB2 processes
    a. Deactivate all databases and stop DB2
    b. Stop administration server
    c. Unload unused shared libraries from memory (*slibclean*)
    d. Stop all instances using DB2 (e.g. Fault Monitor)
    e. Clean all DB2 interprocess communications (IPCs) for the instance to be updated
    f. Verify no DB2 processes are running
6. AIX Administrator - Apply DB2 fix pack image to all servers (apply only; not commit)
7. * AIX Administrator and Database Administrator - Update the instance to the new level of DB2 (*db2iupdt*)
8. * AIX Administrator and Database Administrator – Update the administration server instance (*dasupdt*)
9. Database Administrator - Verify the level of DB2 applied (*lslpp* and *db2level* - this must match the fix pack level being applied)
10. Database Administrator - Start DB2 and the administration server (verify successful startup without errors from db2diag.log)
11. Database Administrator – Connect to each database and rebind bind files (e.g. databases, packages, utilities)

* Note: Some Database Administrator tasks are shared with the AIX Administrator due to root access being required

Validation Testing Tasks

Refer to Appendix C for a detailed discussion of software validation testing.

1. Database Administrator - Verify basic database functionality is working (e.g. administration tools, runstats, selects from partitioned tables, etc.)
2. Application Team - Perform application testing

Appendix C - Building a Software Validation Test Suite

INTRODUCTION

Business users have a common request of their IT organizations - deliver systems that are available when required and consistently provide the expected level of performance. Changes to a system, however, can produce instability, performance degradation and even downtime. This experience can be avoided when organizations establish and maintain a testing program that seeks to discover and address issues resulting from changes before they can effect the production system. As organizations move toward a zero tolerance for instability of the data warehouse, a complete reevaluation of the testing process is required.

WHY DO PROBLEMS OCCUR?

So why do problems occur? In short: for large scale, generalized applications, it is impossible to test for every data combination on every hardware/software platform available. The function may perform correctly in the software developer's test environment, but encounter unexpected or untested conditions when integrated with other software products (e.g. ETL tools), different software release levels (e.g. operating system levels) or hardware combinations. This same experience may result when unique workload conditions exists.

Since software will never be perfect, then organizations must take measures to ensure the stability of their systems. This is not to suggest that software developers should not to strive to deliver higher quality code, but rather to suggest that unforeseen problems can accompany software. Therefore, the need for rigorous testing is critical. All too often, however, new releases of software are promoted to production environment without:

- Thorough testing of basic software functions
- Retesting of previously experienced problems
- Stressing the new software to assess the performance impact of data volume and workload characteristics

WHAT SHOULD THE IT ORGANIZATION DO?

IT organizations that commit to delivering and maintaining systems that are available when users require and meet expected levels of performance will seek ways to identify potential issues early. Since problems can be caused by a number of factors, IT organizations need to address both technical and organizational factors. Consider the following suggestions:

1. Accept complete ownership for the stability of the data warehouse. Change the culture of the organization so every group associated with the data warehouse takes responsibility for the stability of their portion of the environment and participates in delivering the overall stability of the system to users.
2. Standardize the testing processes to maintain the consistency and repeatability of the tests. Establish a validation testing program to discover problems early. Realize that the amount of testing required for new software release levels may be only slightly different, but just as important, from the testing when an application is first deployed. Over time, maintain and enhance the test suite, as appropriate.

3. Create the role of an Application Change Control Coordinator to set the testing strategy and lead the overall testing effort.
4. Determine and adhere to the criteria for how, which, when and where changes will be introduced.
5. Create and track an inventory of current software release levels along with a roadmap for future deployments.

HOW DOES AN ORGANIZATION START?

When building a validation test suite, organizations can borrow strategies and disciplines that have been used for years in the mainframe environment. This section will identify several areas that are helpful to begin or improve a testing program. In summary, organizations need to:

1. Identify everything that needs to be tested
2. Establish the appropriate operational environments
3. Create standardized test suites for functional/performance testing
4. Define the promotion strategy for software changes
5. Have a tested back-out plan

1. Identify everything that needs to be tested

While it sounds fairly simple, identifying everything that needs to be tested involves a lot of work. The following is only a partial list of the broad categories to consider:

- Query workload mix
- Trickle feed process
- Switching views
- Loads
- Backup/recovery
- Maintenance such as Reorgs and Runstats
- Crash recovery
- Apply/revoke permissions

For each of the above categories, several specific tests may be required to validate that software changes continue to provide the functionality or stability of processes in each category.

In addition to the above, a list of every system function (e.g. AIX, DB2) that is important must be identified and added to a checklist. Every time a new release is installed, each item on the checklist needs to be validated to determine that it is working properly. When upgrading to a new software release, any new functions that will be used need to be added to the checklist.

Finally, develop a list of items that will be monitored. While tests are running, items like processor and memory usage, database resources, and I/O and network bandwidth will need to be monitored to compare the difference from one software release level to the next. This ability to assess and compare pre-change and post-change behavior implies maintaining some degree of historical information about the relevant statistics.

2. Establish the appropriate operational environments

For years, three operating environments have been referenced – development, system test and production. What has not been specifically called out, however, is that many organizations deploy multiple test environments. Consider the following possible operating environments.

- **DEVELOPMENT**: Application Development - The Application Development environment is for the ongoing development of applications. Since this environment supports a user population, it is treated as if it was production. In other words, "development" should not be where new software releases are initially installed and tested.
- **TEST**: DBA & System Administration Functional/Performance Test – When new system software needs to be installed and tested, the process begins in the DBA & System Administration Functional/Performance Test environment. In other words, this is the environment where new releases of the operating system and database management system are initially installed. This environment is a scaled down version of the production system and uses static test data. In this way, test results can be compared to previous tests run on older levels of software, but against the same data. Therefore, this environment not only validates that specific functions are working properly, but also validates that performance characteristics have not adversely changed. Since DBAs and System Administrators are the primary users, upgrades to this environment can be done at any time. Further, if changes to this environment introduce any issues, the problem identification and resolution is done without impacting application developers or business users.
- **TEST**: Application Unit Test – After application software code has been developed with some initial testing, the application is promoted from Application Development to the Application Unit Test environment. This environment is treated as if it was production because it has a user population. Having passed the functional and performance testing, new software releases are introduced to variable workload characteristics in this environment. Problems can be discovered in this environment because it more closely mimics the production system. As issues are discovered, the key is to develop a feedback loop so a test is created and included when the next new software release is installed. When the root cause of a problem is not clear, DBAs are good resources to assess the problem and isolate it to application or system software. The Application Unit Test environment is also a scaled version of the production system so the efforts of volume testing with the new software release levels can be assessed.
- **TEST**: System Integration Test – The final test environment for new applications or system software releases is the System Integration Test environment. This environment is treated as if it was production because it has a user population. System Integration Test closely mirrors production in that it regularly experiences data refreshes and mimics the production workload characteristics and mix. Again, it is proportionally scaled to the production system so that the impact of data and workload volume can be assessed.
- **PRODUCTION** – By the time new releases of software reach production, the objective is for any issues to have been identified and resolved. Through a rigorous testing process, the quest to deliver production systems that are available when required by users that meet performance expectations is achievable.

Each organization will have different requirements that will influence the appropriate number of operating environments to establish. The size of the test environments are based on having enough data with a proportional amount of processing capacity to simulate the conditions that will be encountered in production (i.e. all the data types and combinations). For System Integration Test, however, an additional requirement of simulating the volume of workload that the production system encounters needs to be considered. The requirement is to have enough capacity (data and processing) to fully simulate key productions events (e.g. daily/weekly/monthly loads, heavy processing over a given time period) so that they behave the same way they would in production; therefore, the size of the System Integration Test environment varies by organization and is determined by whatever is needed to represent production as closely as possible.

3. Create standardized test suites for functional/performance testing

Without a formal and standardized process, testing will only reflect individual characteristics and experiences of those doing the testing. This will lead to an inconsistent and incomplete testing process. Therefore, testing should be standardized with the creation and maintenance of a test suite. To begin the initial testing of new software releases, the DBA/SysAdmin Functional Test environment should be used. DBAs should have a checklist of key features (e.g. administration commands) that need to be tested. While some of the functions can be scripted, others will require manual testing. Functional testing will continue with a series of predefined test cases that validate proper functioning of tasks that are performed by the data warehouse applications. These tests are created and maintained so that they are repeatable each time a new software release level is installed. In addition to testing the functionality of a product, this process seeks to assess how software components react to each other throughout the software stack. In other words, does the function perform properly based on the specific levels of software installed in the environment and is there any change to the performance characteristics for a given test.

A key role of the Functional/Performance Test environment is to conduct and assess performance testing scenarios. Since standard test data is used, the performance characteristics of the tests can be compared to previous tests to determine if they are still the same. A workload can be established and individual queries measured based on a specific load. Either in-house scripts or actual workload simulation tools can be used to mimic workload volume for a desired period of time. The key to this type of testing is two-fold:

1. Have an environment that is proportionally scaled (i.e. calibrated) and configured (i.e. parameters) like the production system in terms of both processing capacity and amount of data. With this correlation, the test system will react like the production system so the type of adjustments that may be required for production can be evaluated. Software changes can often result in altered system responsiveness that will require tuning adjustments. For example, bufferpools may need to be modified.
2. Create a suite of activities that mimics the workload mix for a given day in production. Application developers can be helpful in determining the workload mix. A prerequisite for this is that the data in the test environment be representative of the conditions that exist in production. Knowing the data is a must to determining how much will be required to create a representation of the data conditions that exist in the production environment.

4. Define the promotion strategy for software changes
New software releases are initially installed in the Functional/Performance Test environment. After passing functional and performance testing, the new software releases can be promoted to the Application Unit Test environment that more closely mimics production. Following a period of time with no issues, the new software release levels can be then promoted to the System Integration environment. This environment introduces many variables; it is never exactly the same because a subset of current production data is used. In this environment, however, very specific and critical tests are performed. For example, large load and trickle feed processes should be tested in this environment. Any process with an element of complexity or variability should be tested in the System Integration environment. Additionally, application programmers use the integration environment and become another source of random workload creation and testing. Again, this environment mimics production workload and is scaled proportionally to the production system based on processing capacity and the amount of data. Changes are ready to be promoted to the production environment when they can run in this environment without incident for up to four weeks.

5. Have a tested back-out plan
In the event that problems cannot be resolved within a predetermined timeframe, have a back-out plan to restore the system to its original state. This plan should include a determination of when to back-out the change and revert to the original state. When making changes, however, always assume a problem is going to occur and include additional time in the maintenance window to resolve it.

HOW CAN ORGANIZATIONS IMPROVE CURRENT TESTING EFFORTS?

The development of testing programs is a process that should be continually improved based on experience. When assessing your testing program, ask these questions:

1. At the completion of installing of a new software release:
- "What could have been done to make this change more stable?"
- "What could have been done to have more thoroughly tested this change?"
2. When problems are found in production:
- "How did the problem get into production?"
- "Why wasn't the problem found in test?"
- "What other tests can be defined and added to the test suite to test for this problem?"

The answers to these questions will lead organizations to:

- Establish a feedback loop to the Problem Management System so a problem is tracked
- Make a commitment to root cause analysis so the real cause of a problem is understood. Often, a unique set of circumstances or specific sequence is required for the problem to occur
- Develop a test case to add to the test suite when the problem is resolved

Identify and resolve issues that are discovered in the test systems. The elusive problems need to be fully investigated. In scaled down test environments, some problems may only surface irregularly, but appear more frequently in a large scale production environment. Defining a scenario to test for these problems is of great value.

While every possible scenario cannot be tested, those that can should have a test that is included in the test suite. This will prevent the same problem from reoccurring when future changes are made. In some cases, however, a certain degree risk must be accepted. A sensible implementation plan will balance the need for diligence with determining what risks can be mitigated and which will be accepted.

WHAT ARE THE BENEFITS TO THE ORGANIZATION?

The stress of problems takes its toll on the business, business users and IT staff. Consider the value when the IT organization delivers a stable system:

- Business user productivity is maintained
- Reputation of IT is enhanced
- Staff productivity increases by avoiding unscheduled support time
- Job satisfaction and quality of life improves with fewer issues to resolve during non-work hours

These benefits are not without cost. Establishing the necessary operating environments can be a significant cost. Staff time to build and maintain the test suites can be a significant cost. These costs, however, must be weighed against the equally significant business cost of instability and downtime.

The effort associated with validation test suites should not minimized. Changes to the organization and established processes will be required and are challenging. The effort to look for root cause analysis requires a commitment of precious time. Thoroughly evaluating new software releases requires a commitment of time while still needing to address other business needs.

In spite of the time, effort and cost to build the test suite, the value of delivering stable systems outweigh the costs. Assess your organization. How much time is spent in unscheduled support activities following software release changes? By tracking unscheduled support time against the time to build and maintain a test suite, some organizations have justified the cost. From a return on investment perspective, it is possible that some organizations will obtain the value at a net zero cost. In other words, they get the value for free. The alternative is to run the risk of production problems where anxious users will be hovering in the office, believing that the world is coming to an end and having no reluctance to make those nighttime phone calls. What is the value of avoiding problems?

CONCLUSION

Delivering systems that are available when required by users that meet performance expectations takes a commitment to invest time, effort and budget. It also takes a willingness to allow more time to get a software change promoted to a production environment; however, a significant amount of wasted time, effort and cost can be avoided when rigorous testing occurs. A process that focuses on avoiding problems by creating a software validation test suite complemented by an organization’s commitment to stable systems will provide substantial and lasting business value.