Enterprise Strategies to Improve Application Testing

Optimized Test Environments Speed the Deployment of Reliable Applications

WHITE PAPER
December 2005
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Improving Application Reliability and Quality</td>
<td>1</td>
</tr>
<tr>
<td>Why is Application Testing a Challenge?</td>
<td>4</td>
</tr>
<tr>
<td>What is Needed to Improve Application Testing?</td>
<td>11</td>
</tr>
<tr>
<td>Princeton Softech Optim Meets the Challenge</td>
<td>13</td>
</tr>
<tr>
<td>Maximize Business Value Across the Enterprise</td>
<td>16</td>
</tr>
</tbody>
</table>

©2005 Princeton Softech, Inc. All rights reserved. Princeton Softech Optim is a trademark of Princeton Softech. All other product and company names are trademarks or registered trademarks of their respective owners.
(Ref. no: 27543-5)
Improving Application Reliability and Quality

Application testing for accuracy, reliability and quality has never been more important. Why? Because companies across industries depend on mission-critical enterprise applications to drive their business initiatives. In turn, these applications rely on relational databases to store and manage the underlying enterprise data.

The ability to enhance, maintain, customize and update these sophisticated applications and complex relational databases is critical for achieving long-term business goals. Companies are striving to speed the deployment of reliable, high quality applications, while staying within tight development budgets.

Now, more than ever, companies face new challenges when designing effective and efficient testing strategies for enterprise applications. Incomplete or flawed test data means inaccurate testing, which can lead to business disaster.

No company wants to risk losing customers, market share or brand equity by delivering applications that have not been thoroughly tested. For this reason, end-to-end application testing is a strategic priority throughout the application development lifecycle.

Enterprise Data Management offers a Competitive Advantage

Enterprise data management includes capabilities for streamlining the total application testing process to be more efficient, repeatable, accurate and cost effective. Improving the way you manage enterprise application data in the testing environment delivers dramatic results including improved reliability, faster time-to-market, reduced development costs and higher quality. In short, enterprise data management helps you do more with less.

Improve Application Reliability

Today’s applications drive revenue and satisfy sophisticated marketplace initiatives. These customer-facing and mission-critical applications have a huge strategic impact. As a result, there is increasing focus on application reliability to minimize unplanned downtime and to engender customer loyalty.
According to *Computerworld*, “buggy software is costing the economy billions of dollars.”\(^1\) From inaccurate election results to drastically miscalculated airline fares to the largest power outage in North America in August of 2003, unreliable application software can lead to unrecoverable, costly downtime, and the problem is getting worse.

Software defects directly affect a company’s bottom line. Failure to properly test applications through the various development phases is recognized as the primary cause. Can your business sustain the revenue impact and customer loss that could result from an application failure?

**Speed Time to Market**

Delivering quality applications in a timely manner provides companies with a competitive advantage that can improve revenue. Shorter development cycles are desirable, but quality testing takes time. This classic struggle between these two business objectives often means that release dates slip. How important would it be to your business if you could get applications to market faster without sacrificing quality?

**Control Development and Quality Assurance Costs**

Reducing the overall cost of developing, testing and delivering reliable applications can result in significant cost savings throughout the entire application development lifecycle. Companies can realize direct cost savings and productivity gains by reducing the work effort required by DBAs, developers and quality assurance testers, allowing them to address other business issues. Could your business benefit from reducing development costs and using talented IT resources more effectively?

The costs associated with resolving application defects in the production environment are as much as 10 to 100 times greater than if these defects are resolved in the development process. Even worse, when your customers are the first to discover application defects, they become less “loyal” to your company and your products. Could your company afford the increased direct expenses and indirect opportunity costs of releasing faulty software?

**Lower Infrastructure Costs**

In application development and testing environments, there are often several clones of the original production database. How many database copies do you use? It is important to consider that as data is duplicated, costs increase proportionally. Using right-sized databases rather than clones can help you save or reclaim storage assets. In addition, by reducing the size of test databases you will also reduce the processing power needed to perform testing activities. Could your business benefit from using smaller right-sized test databases?

---

Delivering Reliable Applications

Development organizations realize that more planning is needed to satisfy the rigorous testing objectives necessary to deliver reliable applications. Designing a comprehensive testing strategy is often taken for granted and yet, can be as challenging as designing the application. Organizations also recognize the need for proven test data management and automated testing software to manage the quality assurance process cost effectively. Application testing is fast becoming a high priority because resolving problems early in the development stages can significantly reduce costs.

Many organizations are reluctant to make a significant investment in testing, allocating the money to other areas. Clearly, quality improvements carry a cost. However, without a commitment to quality assurance, the business consequences could be significant. It is necessary to understand the impact of poor application quality and reliability on critical business goals, such as generating revenue and ensuring customer satisfaction.

Investing in your development projects by using effective enterprise data management capabilities and repeatable testing processes will ultimately increase your return on investment. In today’s market, quality is a major competitive differentiator. If companies want to be competitive, application quality and reliability must be a high priority.
Why is Application Testing a Challenge?

Typically, each time a new application is developed, or an existing application is modified, a new test database is created. Since it is ideal to use “realistic” test data, the test database is usually a clone of the production database. Special test cases may be added before testing begins.

Iterative testing involves executing the application using the test database and verifying the results to ensure the application is working as designed. Any problems discovered must be resolved, and the test data must be refreshed, before testing continues.

This process is repeated throughout the various testing phases (unit, integration, system, load, regression and acceptance testing) until the application is migrated into production. In addition to finding and resolving application errors before deployment, another important goal is to create a repeatable testing process that improves application quality, reduces time to market and minimizes costs.

Complexities of Testing with Relational Data

The fact that most applications rely on relational database technology introduces a major challenge for organizations in the testing process. The application data model may contain dozens, hundreds or even thousands of tables, with just as many interrelationships. Data model complexity is not limited to large-scale systems. Even a database of less than a dozen tables can contain relationships that make navigating the data model difficult.

Without the appropriate solutions, developers often need to write sophisticated extract programs just to create the test data. It is a challenge to navigate the numerous tables, rows and columns to create, manipulate and refresh the desired database subsets. It is also difficult to develop an extract program that considers relationships defined and enforced by the application. Finally, whenever the application or database is updated, the extract program must be updated as well.

After a test is executed, there is still no clear and simple method to verify that the results are valid. Adding to the complexity is the challenge of handling heterogeneous database management systems and differing data models. Clearly, comprehensive testing capabilities must support these relational complexities and “remember” to account for them in every extract, compare or update operation.
Test Data from Multiple Databases

Most organizations have data stored in a variety of relational databases, such as Oracle, DB2, DB2 UDB, SQL Server, Sybase and Informix. In addition, data may be stored in hierarchical or non-relational formats, such as VSAM files and IMS databases. When testing complex applications, it is not uncommon to require test data from multiple related databases including both relational and non-relational data sources.

All database management systems have different methods for handling data. The integration of diverse systems is a typical requirement. Quality testing must also include seamless capabilities to handle data from different database management systems operating on different platforms.

Test Data from Homogeneous Databases

In homogeneous database environments, the technology is the same or compatible. One of the challenges of creating test data in a homogenous database environment is the ability to extract complete subsets of related data and to keep that data referentially intact. It is necessary to navigate all relationships in the data model, whether they are defined in the database or the application (see Figure 1).

Test Data from Heterogeneous Databases

Heterogeneous database environments can include different hardware, operating systems and data models, as well as syntactic and semantic differences. Creating test data from heterogeneous databases adds data compatibility and translation issues. Typically, this happens when moving an application to a different database or test environment. For example, it may be necessary to extract subsets of data from a DB2 UDB production database and insert the data into an Oracle test database. This transition may cause problems with special registers or date and time stamps, which are handled differently by each DBMS (see Figure 2).
Beyond extracting referentially intact subsets of data, when working with heterogeneous databases, you need additional capabilities for managing compatibility and translation differences automatically.

**Federated Test Data from Multiple Relational Databases**

Federated environments often require extracting related test data stored in multiple relational databases, which adds another level of complexity. For example, payroll data may be managed in a DB2 UDB database and employee data in an Oracle database (see Figure 3).

![Federated Relational Test Data](image)

In addition to maintaining referential integrity and handling data compatibility and translation issues, creating realistic test data now requires a federated data access capability, so you can extract related data from multiple databases using a single extract process.

**Federated Test Data from Non-Relational and Relational Test Data**

While relational databases have inherent complexities, the task of testing an application that references both non-relational and relational data poses an even greater challenge. For example, an order entry application may require product data from a VSAM inventory file and customer data from a DB2 order entry table. Data stored in VSAM files or IMS databases have different structures and cannot easily be integrated with relational data (see Figure 4).
An effective testing strategy in these scenarios must include the ability to create precise subsets of a realistic federated test data from non-relational and relational databases. Because the database structures are significantly different, it is a major challenge to correlate data across complex non-relational and relational sources. What is needed is a tool that accesses VSAM and IMS data and correlates that data in a way that can be easily “understood” and managed.

Because VSAM and IMS databases are less flexible, it is difficult to extract subsets of data from these databases without the ability to specify simple or complex selection criteria. The ability to apply selection criteria dynamically to extract VSAM or IMS data would significantly reduce processing overhead and improve the quality of the test data.

**Building a Test Database**

Some of the more common approaches to building test databases include cloning the production database(s) and writing custom programs to extract subsets of test data. However, these methods are labor intensive, error prone and require extensive CPU and network resources.

Typically, it is impractical to clone an entire production database comprising hundreds of highly interrelated tables just for testing purposes. First, there is a capacity issue. Second, there is a quality issue — when working with large test databases, developers may find it difficult to track and validate specific test cases.

Cloning an entire production database increases the time needed to run test cases because there is a larger volume of data. In addition, the production data may not contain the specific test cases required for effective testing. It is much faster to test with smaller, realistic subsets that accurately reflect the production data, without adding overhead to the testing process.
At the other end of the spectrum, some IT organizations rely on “toy” test databases that have been cobbled together by hand or populated using custom coded extract programs. Usually, these databases fail to reflect the actual complexity and breadth of their production counterparts. The applications are tested and put into production only to break down when they encounter “live” processing conditions.

**Creating Test Data**

When the test data is based on an existing production database, extract programs are used to copy data from the production database to populate a test database. Some techniques for identifying and creating test data include selection criteria, random selection, data partitioning/grouping, or limiting the data by table or relationship. However, without a generalized solution, all of these methods involve writing custom programs.

**Referential Integrity Rules**

Referential integrity (RI) rules may be enforced by the database, by the application, or both. Typically, the application-enforced RI is more complex. For example, the application may include relationships that use compatible but non-identical data types, composite and partial columns, and data-driven relationships. The extract program must have capabilities for handling every type of relationship.

**Multiplying the Test Data**

Often when creating new test data for individual or multiple users, a small set of the “right test data” is “multiplied” into additional sets of data. The requirement to synthesize test data sounds easy, but has many challenges.

First is the need to modify primary key values to prevent creating duplicate rows. Next, the modified primary key value in a parent (or owning) table must be propagated to all child (or subordinate) tables to maintain the referential integrity. Without the capability to propagate data and maintain referential integrity accurately, there is no way to guarantee the quality of the test data.

**Forcing Error Conditions**

Creating realistic subsets of related test data from a production database is a reasonable start, but it is sometimes necessary to edit the data to force specific error conditions or to validate specific processing functions. A comprehensive relational editor would not only provide the capabilities necessary to create this special data, but would then make it easy to browse data and resolve problems. The ability to browse data in its relational or business context also provides a clear way to envision the data relationships and structure of the data model.
Transform and De-Identify Test Data

With the increased focus on data privacy, the ability to transform and de-identify sensitive data is critical for avoiding severe penalties. However, given the complexities of relational data, masking sensitive data is not an easy task. What is needed is a flexible way for mapping source and destination columns and capabilities for using a variety of transformation functions and algorithms.

For example, it would be easy to de-identify customer identification numbers or social security numbers by simply applying a random number function. More sophisticated capabilities would allow you to use substrings, sequential numbers, date aging, currency conversion, or table lookup functions. The capability for including user-defined data transformation programs would provide even greater flexibility to satisfy complex or site-specific requirements.

All of the methods described are effective techniques for de-identifying test data. However, in testing relational database applications, there is an added complication. Specifically, the tester requires the ability to propagate a masked data element in a parent table to all of the related child tables in the database. Key propagation is necessary in order to retain the referential integrity of the transformed data. Otherwise, the relationship between parent and child tables is severed, test data is inaccurate, and application testing yields invalid results.

Key propagation also provides the ability to “manufacture” new sets of test data. This technique is helpful when additional data is required for testing, as is the case when testing new applications or testing applications when new tables are added to the data model.

Validating the Test Results

Without the capability to automatically compare the test data before and after a test run, validating test results and identifying changes are next to impossible. First, there are a variety of changes: inserts, deletes and updates spread across hundreds of tables. Second, there may be unexpected problems (for example, orphaned rows) and other anomalies that may go undetected if the results are analyzed manually.

An effective testing strategy must compare subsets of related data and identify the differences automatically. With relational data this means more than comparing row to row. It means using data model intelligence to compare related sets of rows.

Maintaining Your Test Environment

The quality of the test environment is just as critical as the quality of the test data. The content of the test database is modified during the testing process. Over time, it diverges further from the baseline test data, resulting in a less than optimal test environment. Without capabilities for saving and reusing the test data, it is more difficult to refresh the test database to ensure accurate results on subsequent test runs.
Unless the extract process includes metadata, there would be no way to accommodate changes in the data model during the testing phases. Metadata is definitional data that provides information about the structure of the data managed within an application or environment. For example, metadata would document the structure of the database including the tables, columns, relationships, views, triggers and so on.

Working with predefined realistic subsets of data and refreshing the test data iteratively throughout each test cycle improves testing and overall application quality. Including metadata in extract processing ensures that you can recreate the test data and accommodate data model changes quickly and accurately. The ability to save and reuse processing specifications streamlines the testing process and ensures a consistent and manageable test environment.
What is Needed to Improve Application Testing?

As long as there are application modifications and enhancements, testing is necessary throughout the application development lifecycle. Each phase, from unit testing through system integration and acceptance testing has unique requirements and varying levels of complexity. So what is needed to improve application testing?

Test data management and automated testing capabilities that help you improve every phase in the application testing process. You save time, ensure accuracy, speed time to market and reduce costs. Most importantly, you have the flexibility to plan and implement effective testing strategies to ensure the reliability of the most complex applications.

Selecting Comprehensive Testing Capabilities

The commitment to deliver thoroughly tested applications and still reduce costs requires comprehensive test data management capabilities that enable you to:

- Create realistic, referentially intact subsets of production data from multiple data sources for accurate and efficient testing.
- Support the most complex data models and navigate relationships, whether defined by the database, the application or both.
- Save, reuse and share processing specifications to refresh test databases quickly and accurately to maintain a consistent test environment.
- Browse and edit test data in its relational context to create special test cases and force error conditions.
- Compare the before and after test results automatically to identify problems and anomalies that might otherwise go undetected.
- Integrate automated testing with test data management to realistically test application functionality with complete data accuracy.

These capabilities should be readily available to all members participating in the development and testing process to ensure a fast, easy and resource-efficient test environment.
Managing the Test Environments

Building and populating realistic test databases requires the capability to extract and move precise subsets of related data based on your specifications. Maintaining referential integrity with 100 percent accuracy is essential for the simplest and most complex data models. In addition, capabilities for saving and reusing extract, insert and load specifications save time in refreshing test databases and ensure a consistent test environment.

Comprehensive capabilities for transforming and de-identifying test data are required. This added flexibility allows for creating special test data and is especially important to comply with government regulations for protecting privacy.

Creating test data to force error conditions requires intelligent browsing and relational editing capabilities. Relational browse features improve the ability to perceive data relationships clearly and to identify and resolve problems quickly. The ability to insert rows and edit database tables directly improves productivity and accuracy. Multi-level undo capabilities are essential.

The ability to identify data anomalies and inconsistencies during testing is essential to the overall quality of the application. The only way to truly achieve this goal is to deploy an automated capability for comparing the before and after images of your test data. Speed and accuracy are essential. Automated compare processing saves time and ensures that you can identify problems that would otherwise go undetected.

Automated Testing

Developing an effective test strategy requires both automated testing processes and test data management. Automated testing emulates user interactions that exercise and test application functionality. Test data management provides appropriate and realistic test data to ensure that automated test scripts exercise all paths of application logic and generate reliable results.

Any automated testing process that does not include test data management is flawed. Using incomplete or inaccurate test data may generate incorrect or unreliable test results. When deployed together, automated testing and test data management provide a powerful, efficient combination that increases the value of automated testing.

Scalability

How well does the testing strategy work when the application testing requirements expand? Application development and testing environments are dynamic. Test data management capabilities must adapt to any arbitrarily complex data model for large and small databases; otherwise, the IT organization will have extra work to do. Any software technology that claims to support application testing must not impose artificial limits on the number of tables or the kinds of relationships. Any limitations place the IT organization back into the custom coding business.
Princeton Softech Optim Meets the Challenge

Princeton Softech Optim™ enables IT organizations to meet even the most complex application testing challenge by providing all the fundamental components of an effective test data management strategy:

- Extract referentially intact subsets of data with 100 percent accuracy to create realistic test databases no matter how many tables or relationships are involved.
- Insert or load subsets of related data to quickly build realistic test databases. Update or refresh the test data consistently to preserve the integrity of the test environment.
- De-identify sensitive data in the test environment to ensure compliance with regulatory requirements for data privacy. Transform test data to meet specific test case requirements.
- Browse and edit test data to force error conditions and resolve problems. Reviewing data in its relational business context provides a clear vision of the data model.
- Compare the test data before and after exercising the application to validate expected test results and identify anomalies automatically and with pinpoint accuracy.
- Integrate test data from multiple related databases and database management systems (Oracle, DB2, DB2 UDB, Sybase, SQL Server and Informix).
- Integrate relational and non-relational data to create a federated test environment.
- Integrate test data management with automated testing to provide comprehensive testing capabilities.

When these capabilities are in place:

- Developers can ensure that new application functions perform as expected during unit testing and modifications do not introduce problems during integration testing.
- Quality assurance staff can ensure the entire system operates as expected and validate that interfaces with other systems work properly.
- Business-unit users can ensure that the system meets their expectations for functionality and performance during acceptance testing.
- DBAs can spend less time on extracting data to create and maintain the test environment and more time on performance tuning, backup and recovery processing and managing production databases.
What Makes Optim Unique?

Princeton Softech’s Relationship Engine™ is a unique technology that understands and processes related data from multiple tables and ensures that each test subset is always referentially intact and logically complete.

For example, in a subset of customer data, one customer may have items that are backordered while another may not. From one customer to the next, the number of rows retrieved from any number of tables will vary. But the Relationship Engine always gets the right items for the right customers, complete and intact, every time.

Optim uses an active repository to store the user-defined business rules (see Figure 5). The repository automatically captures relationship information defined to the database. Additional data model information can be populated from third party dictionaries or data modeling software. Finally, users can define relationships that exist but are not known to the DBMS, such as application-managed relationships.

Using the active repository, IT staff members can define, share and reuse different Access Definitions that specify, in essence, varying subsets of the database — related rows from many tables. Princeton Softech’s Relationship Engine technology assembles all related rows from every table into a single referentially intact subset.
What Does Optim Offer?

Optim capabilities allow users to migrate, browse, insert, load, transform, edit and compare complete subsets of related data. Organizations can design comprehensive testing strategies that include realistic test data while improving productivity and overall application quality and reducing development costs (see Figure 6).

![Diagram of Optim's process]

**Figure 6.** Optim improves every stage of the application testing process.
Maximize Business Value Across the Enterprise

You invest millions in your enterprise applications and the supporting infrastructure to ensure optimal operating performance, improve decision-making and gain a competitive advantage. Princeton Softech Optim provides the power of enterprise data management, so you can derive the most business value across your enterprise.

Implement a Consistent Data Management Methodology

Princeton Softech Optim provides a single solution for managing enterprise application data throughout every stage of the information lifecycle. Now you can assess and classify application data by age and status. You can apply business rules to archive, subset, access, store, retain and protect your enterprise data. Optim capabilities are based on a consistent and proven data management methodology that aligns with your business objectives and scales across applications, databases, operating systems and hardware platforms.

Speed application deployment by streamlining the way you create and manage test environments. Subset and migrate data to build realistic and “right-sized” test databases. Eliminate the expense and effort of maintaining multiple database clones. De-identify, mask and transform confidential data to protect privacy and minimize the potential for unauthorized disclosure.

Optim supports all major enterprise databases and operating systems, including Oracle, DB2 UDB, Sybase, SQL Server, Informix, DB2, IMS, VSAM, Windows, Unix, Linux and z/OS. And it supports the key ERP and CRM applications in use today — Oracle® E-Business Suite, PeopleSoft® Enterprise, JD Edwards® EnterpriseOne, Siebel® and Amdocs® CRM, as well as your custom and packaged applications.

Optimize Performance, Mitigate Risks, Control Costs

Exploit the power of Enterprise Data Management to realize measurable business value across your enterprise:

- Align application performance to business processes and profit from superior performance and availability
- Simplify database administration, ensure business continuity and speed disaster recovery
- Streamline application and database upgrades and reduce resource requirements for key IT operations
• Automate data retention to support compliance initiatives and respond quickly and accurately to audit and discovery requests

• Leverage existing investments in applications, databases and storage and eliminate IT budget variances

Optim is designed to ensure successful implementation. Princeton Softech’s Enterprise Data Management experts work with your project team to define business objectives and processes for managing enterprise application data in your environment. After implementation, you can count on Princeton Softech for ongoing optimization and technical support.

**Princeton Softech: The Proven Leader in Enterprise Data Management**

Princeton Softech provides enterprise solutions which align application data management with business objectives. Our industry leading Optim solution enables organizations to optimize performance, mitigate risks and control costs. Partnered with the market leaders in business technology, we deliver capabilities that scale and support your enterprise – applications, databases and platforms. More than 2,200 companies worldwide – including nearly half of the Fortune 500 – rely on Princeton Softech’s proven solutions to maximize the business value of their enterprise applications and databases.