Part 12 of 20

Testing: The Quality Discipline That Protects the Go-Live

The testing hierarchy, test script development, defect management, performance testing, and the go-live readiness criteria that make testing rigorous rather than ad hoc

CONTENTS OF THIS PART

WHAT YOU WILL LEARN AND WHY IT MATTERS

Testing is the quality discipline that protects the go-live decision — the organizational commitment that the new financial system is ready to replace the legacy system as the financial system of record. A go-live executed on a system that has been inadequately tested is a go-live that accepts unknown risk: the defects, configuration errors, and integration failures that testing would have identified before go-live will be discovered in production, when they affect real financial records and real business operations rather than test data in a controlled environment.

The consequences of inadequate testing are asymmetric: the cost of additional testing before go-live is incremental — a few weeks of additional timeline and testing resource cost. The cost of critical defects discovered in production is potentially severe — corrupted financial records, failed audit controls, disrupted business operations, and the management attention required to diagnose and remediate production issues while simultaneously running the business on an unstable financial system.

This part covers the complete testing framework for ERP implementation: the four-level testing hierarchy, the test script development discipline that makes testing rigorous, the defect management process that tracks and resolves issues systematically, the performance testing that validates system behavior at production scale, and the go-live readiness criteria that provide the analytical foundation for the go-live decision.

THE TESTING HIERARCHY

The testing hierarchy organizes the testing effort into four levels, each building on the results of the previous and each addressing a different dimension of system quality. Moving through the hierarchy in sequence — not advancing to the next level before the current level is complete — is the most important structural discipline in the testing program.

Unit testing is the first level and the most technically detailed. It verifies that individual system components — specific configuration settings, individual workflow steps, specific calculation rules — produce the correct results when tested in isolation. Unit testing is typically performed by the implementation partner's configuration team as each component is completed during the build phase, and it is the quality gate that ensures the individual building blocks of the system are correct before they are assembled into the complete process flows tested at higher levels.

Integration testing is the second level and the first level at which the complete end-to-end flow of a business transaction is validated. An integration test for the order-to-cash process follows a test customer order from creation through revenue recognition to AR recording — verifying that data passes correctly between modules, that each module performs its required calculations, and that the cumulative financial impact of the transaction is correct. Integration testing also validates the external system connections, verifying that data flows correctly between the ERP and each connected system. Integration testing is

typically performed jointly by the implementation partner and the client's internal project team.

User acceptance testing is the third level and the level at which the business stakeholders — the actual users of each process — validate that the configured system meets their requirements. UAT is performed by the functional users rather than by the implementation or testing team, because the purpose of UAT is to verify that the system meets the business requirements from the perspective of the people who will use it, not to verify that it meets the technical specifications of the configuration team. The UAT test cases should be derived from the business scenarios in the requirements document — the specific business situations the system must handle correctly — rather than from the technical configuration specifications.

The parallel run is the fourth level and the most realistic test of system quality. During the parallel run, both the legacy system and the new system process the same transactions for a defined period — typically one to two monthly close cycles — and the results of both systems are compared. Discrepancies between the two systems reveal configuration errors, mapping failures, and calculation differences that did not surface in the earlier testing levels. The parallel run is the highest-confidence validation available for the financial record accuracy of the new system, because it subjects the system to actual production transactions rather than carefully constructed test cases.

TEST SCRIPT DEVELOPMENT

Test scripts are the documented instructions that specify exactly what steps to perform in a test, what data to use, what results to expect, and how to evaluate whether the test has passed or failed. They are the foundation of rigorous testing — the discipline that transforms testing from an informal system exploration into a structured quality assurance process with verifiable, auditable results.

A well-written test script has four components. The objective describes the specific system behavior or business requirement being tested — what the test is designed to verify. The preconditions specify the system state and data conditions that must exist before the test begins — the specific customer records, vendor records, or configuration settings that the test depends on. The test steps provide the detailed sequence of actions to be performed, including the specific menu paths, field entries, and system functions to be used at each step. And the expected results specify exactly what the system should produce at each step and at the end of the test — the specific screen displays, the specific calculated values, the specific records created or modified.

The expected results specification is the most important component of a test script because it defines the criteria for test passage. A test script without expected results produces a testing exercise that can verify that the system does something but cannot determine whether it does the right thing. Expected results must be defined before the test is executed — not derived from what the system actually produces during the test, which simply confirms the current behavior rather than validating it against requirements.

Test case coverage is the metric that measures the completeness of the testing program — the percentage of business scenarios documented in the requirements that have a corresponding test case. For must-have requirements, test case coverage should be one hundred percent: every must-have requirement should have at least one test case that verifies its correct implementation. For should-have requirements, coverage should be at least eighty percent. For nice-to-have requirements, coverage should be proportionate to the criticality of the requirement, ranging from fifty to one hundred percent depending on the financial impact of a defect in the requirement.

DEFECT MANAGEMENT

Defect management is the process for tracking, prioritizing, and resolving the issues identified during testing — the systematic approach that ensures every identified defect is recorded, assessed for priority and impact, assigned to an owner, and resolved before go-live. Without a disciplined defect management process, the testing effort produces a backlog of identified issues that are not systematically tracked, not consistently prioritized, and not reliably resolved before the go-live decision.

The defect log is the central artifact of defect management. Every issue identified during testing should be logged immediately with a standard set of information: the defect identifier, the description of the observed behavior, the expected behavior based on the requirements, the severity classification, the steps to reproduce the defect, the module and process affected, and the owner responsible for resolution.

The severity classification is the most important element of the defect record because it determines the priority of resolution and the impact on the go-live decision. Critical defects are those that prevent a required process from functioning correctly and have no acceptable workaround — they must be resolved before go-live. High severity defects are those that significantly impair a required process but have a temporary workaround available — they should be resolved before go-live, but a documented workaround may allow go-live to proceed if resolution is not feasible within the available timeline. Medium severity defects are those that impair non-critical processes or that have adequate workarounds available — they should be tracked and resolved in the first post-go-live enhancement cycle. Low severity defects are cosmetic or minor usability issues — they can be addressed in subsequent enhancement cycles.

The go-live defect threshold is the maximum number and severity of open defects that the organization will accept as a condition of proceeding with go-live. The threshold should specify separately the acceptable number of critical defects — which should be zero — high severity defects — which should also be zero or very close to zero — and medium severity defects — which may be accepted with documented workarounds and a committed resolution timeline. Defining the go-live threshold before testing begins prevents the threshold from being relaxed under go-live pressure in ways that accept genuine production risk.

PERFORMANCE TESTING

Performance testing validates that the ERP system responds within acceptable timeframes at the transaction volumes and user concurrency levels expected in production — not at the low volumes used in functional testing, where performance is typically not a constraint. Performance problems discovered in production — transaction processing that takes thirty seconds instead of three, report generation that times out under concurrent user load — are among the most disruptive and most difficult to resolve post-go-live issues because they affect every user simultaneously and require architectural changes rather than configuration corrections.

The performance testing scope should include the system functions that will be used most intensively in production and whose performance most directly affects user productivity. For a typical subscription software company ERP, the highest-priority performance test scenarios include the monthly close journal entry processing — the large batch of automated and manual journal entries that must complete within a defined window during the close — the month-end report generation for the management reporting package — the production of the complete suite of management reports from a full month's transaction data — and the concurrent user load test — the simulation of the peak number of simultaneous users performing normal system functions.

The acceptance criteria for performance testing should specify the maximum acceptable response time for each tested scenario at the defined user load. For interactive transaction processing — order entry, invoice approval, journal entry posting — user experience standards typically require response times below three to five seconds. For report generation — the production of complex management reports from large datasets — acceptable response times vary by report complexity and user tolerance, but reports that take more than sixty seconds to generate will generate user resistance. For batch processing — the automated journal entries, the overnight data loads, the period-end close processes — the acceptance criterion is that the process completes within the defined maintenance window without overlap with normal business hours.

GO-LIVE READINESS ASSESSMENT

The go-live readiness assessment is the formal evaluation of whether the ERP system, the organization's processes, the data migration, and the user training are collectively ready for the system to become the financial system of record. It is the decision gate that protects the organization from a premature go-live — a go-live executed before the system, the data, or the users are genuinely ready — which is among the most damaging events that can occur in an ERP implementation.

The readiness assessment should evaluate eight dimensions, each against specific acceptance criteria defined before the assessment begins. Testing completion assesses whether all test levels have been completed with acceptable defect closure. The acceptance criterion is that all critical and high severity defects are closed, and the number of open medium severity defects is below the defined threshold. Data

migration validation assesses whether the migrated data is complete, accurate, and consistent against the validation criteria defined in Part Eight. Training completion assesses whether all users have completed the required training with acceptable competency assessment scores. Integration testing assesses whether all required integrations have been tested end-to-end and are operating reliably. Change management readiness assesses whether the organizational change management activities — communication, training, super-user preparation — have been completed for all stakeholder groups. Cutover planning completion assesses whether the detailed go-live cutover plan — the step-by-step sequence of activities for the transition from legacy to new system — is complete and has been reviewed by all stakeholders. Hypercare planning completion assesses whether the post-go-live support structure is in place — the support team staffing, the escalation path, the monitoring plan. And rollback planning completion assesses whether the contingency plan for reverting to the legacy system — the conditions that would trigger a rollback, the steps required to execute it — is documented and ready to execute.

The go-live decision should be made by the executive sponsor based on the readiness assessment results, not by the project team or the implementation partner. The executive sponsor has the organizational authority to hold the go-live date if the assessment reveals material readiness gaps, and the accountability for the business consequences of a go-live decision made in the face of known unresolved risks.

ACTIONS TO TAKE IN THE NEXT THIRTY DAYS

The following actions will build the testing foundation for a rigorous ERP quality assurance program.

The first action is to develop the test plan — the document that specifies the testing approach, the testing levels, the test case coverage targets, the defect management process, and the go-live readiness criteria — and share it with the implementation partner, the internal project team, and the executive sponsor before the build phase begins. The test plan establishes the testing expectations for all parties and prevents the testing phase from being compressed when build delays create schedule pressure.

The second action is to identify and recruit the user acceptance testing team — the business stakeholders who will perform UAT — and begin their engagement with the project team during the design phase. UAT participants who have been involved in the design phase understand the system design rationale and can produce more targeted and more effective UAT feedback than those who encounter the system for the first time during UAT.

The third action is to develop the go-live readiness criteria for each of the eight dimensions described in this part, specifying the specific acceptance criteria — the defect counts, the test completion percentages, the training completion rates — that must be achieved for each dimension before the go-live decision can be made. The criteria should be developed collaboratively with the implementation partner and reviewed by the executive sponsor before the testing phase begins.

The fourth action is to assess whether a formal parallel run is feasible for the implementation, and if so, to plan the parallel run timing — ideally the month-end close immediately preceding the target go-live date — and the resource requirements. The parallel run requires that both the legacy system and the new system process the same transactions simultaneously, which doubles the finance team's workload during the parallel period. Assessing the resource availability for the parallel run before committing to it prevents the parallel run from being cancelled under workload pressure when it would be most valuable.

CLOSING PERSPECTIVE

Testing is the investment in production quality that most organizations make inadequately and all organizations benefit from making well. The difference between a go-live that succeeds — where the system performs correctly, the data is accurate, and the users are effective — and one that struggles is frequently the difference between rigorous testing and sufficient testing. Rigorous testing finds the defects that sufficient testing misses, and finding defects before go-live costs significantly less than finding them in production.

The CFO who insists on rigorous testing — who refuses to compromise the testing program under go-live schedule pressure, who holds the go-live threshold firm against the organizational momentum to proceed, and who makes the go-live decision based on readiness evidence rather than timeline pressure — is exercising the governance authority that protects the organization from the most common and most recoverable form of ERP implementation failure.

COMING NEXT IN THE SERIES

Part 13 — Go-Live and Stabilization: The Critical First Ninety Days

Part Thirteen covers the go-live event and the hypercare period that determines whether the implementation stabilizes successfully — the go-live day sequence, the first month-end on the new system, the common failure modes and real-time responses, the stabilization milestones, and how to communicate with the board and investors through the go-live transition.