



Managing Quality in your ERP Project: 12 Mistakes to Avoid & Best Practices to Adopt

White Paper

Dan Downing
Mentora Group
www.mentora.com

Abstract

Ensuring quality in an ERP implementation is a major team effort – by your implementation partner, hosting provider, software vendor, and business and technical teams. Testing packaged applications has unique challenges and risks. Unlike with in-house applications, you need to verify the quality assertions of your ERP vendor and prove the robustness of your hosting provider's infrastructure. While you rely on your implementation partner's application experience, don't leave testing to them. Doing so can result in inadequate coverage and insufficient metrics for basing business decisions. Testing should be a checks-and-balances subproject, and be separately staffed. It should report progress *and* quality with high visibility. Apply automation tools selectively or you will be doomed to manual testing for the life of the system.

This paper shines the spotlight on key mistakes that ERP implementations should avoid. They've been distilled from multiple ERP testing projects as part of global enterprise implementations.

About the Author

Dan Downing is the Vice President of Testing Services at Mentora Group. He is a 25-year veteran of the technology and software industry, and has held several executive management positions, and built two consulting practices on application performance and functionality testing.

Dan is the author of the *5-Steps of Web Stress Testing*, and has taught stress testing methodology and led load testing projects for the past 9 years. He is a regular speaker at Mercury and STAR conferences.

Table of Contents

Abstract	2
About the Author	2
How Managing Quality in ERP Projects is Different	4
Competing Stakeholders	6
12 Quality Risks	8
# 1 – Confusing product quality with implementation quality	9
# 2 – Trusting your hosting provider’s assertions.....	11
# 3 – Failure to secure strong sponsorship for testing	13
# 4 – Confusing <i>conference room pilot</i> with <i>test case development</i>	14
# 5 – Believing that testing can be done by the implementation team	15
# 6 – Failure to allocate enough time for testing	16
# 7 – Failure to make quality visible early-on	17
# 8 – Failure to use the right tools to support testing.....	19
# 9 – Believing that automated testing replaces manual testing	21
# 10 – Failure to secure enough testing infrastructure.....	22
# 11 – Underestimation of the effort needed to create reusable test data	23
# 12– Believing that you’re done with testing once you go live	24
Summary	25
About Mentora Group	26

How Managing Quality in ERP Projects is Different

For this discussion, I define ERP application as any purchased package on which your organization runs a significant part of the business. The common ones include one or more of these:

- Financials
- Manufacturing and Distribution
- Supply Chain and Warehousing
- Human Capital Management
- Sales and Customer Relationship Management

In healthcare companies, there are integrated healthcare, customer service, and hospital management packages; in higher education, student administration an enrollment; in financial services, brokerage and trading systems.

You may have other examples in your business. Whichever applies to you, unlike applications developed in-house, packaged ERP applications share common traits and present unique challenges and quality risks that you need to recognize and apply strategies to mitigate.

These unique challenges can be categorized as below.

Enterprise scope

Procuring and implementing an ERP package is an expensive and high-intrusion effort. There are many different stakeholders each competing for the same scarce resources. We will develop this in more detail shortly.

Application Configurability

Much of the appeal and power of packaged software derives from its ability to be configured to an enterprise's specific business processes, and interfaced to other surrounding systems that feed it or are fed by it.

This configurability, however, raises the quality bar —every implementation is unique, and testing must test the specific workflows, business rules, optional fields, converted data that the business configuration process has defined.

Vendor Dependency

Unlike applications developed in-house, you don't have the source code, your business users and analysts are unfamiliar with the functions and usage of the package, and you don't control the maintenance schedule. You are dependent on the ERP vendor for all of this. And thus you have little control of the quality of the software.

Technology Stack

Packaged applications have evolved over the last decade or longer. As technologies have evolved in parallel, these packages have incrementally evolved to leverage this technology, resulting in some cases in hybrid client server / web systems with a mix of standard and proprietary communication protocols.

Occasionally vendors have re-architected their systems from scratch to overcome the complications and performance issues that evolution / hybridization can create.

But in either case, the layers of web servers, java application servers, clustered database technologies, middleware, object libraries, browser plug-ins, data mining options, etc., make for very complex technology stacks. It requires a good deal of know-how to implement them correctly, involving several deep disciplines, and even more expertise to configure and tune them for performance in *your* hardware environment.

Competing Stakeholders

The enterprise scope of an ERP implementation engages multiple stakeholders, both inside and outside of your organization. While all must come together as partners to deliver a successful implementation, each has a different role, different motivations, sometimes conflicting personal agendas, which often end up competing for the same scarce resources of time, budget, business benefit, and personal achievement.

These competing stakeholders typically are:

- **ERP vendor.** They will assert that the quality of their software is better than ever (as long as you are on the latest maintenance release, service pack, and patch, of course). Industry analysts disagree: they see a worsening, not an improving ERP quality trend. And if you've implemented packaged software before, you know that keeping up with maintenance patches without introducing more risk is a tricky process.
- **Implementation Consultant.** Companies engage a consulting company knowledgeable in the specific ERP package to manage the complex implementation activities: workflow configuration, data conversion, building system interfaces, customizing reports, project management. To maximize control, the consultant will include testing in their plan – but assume it will be done by the same business users that are involved full-time in configuring. While “user acceptance testing” of end-to-end processes is vital, it does not replace the more rigorous functional testing of the configured application, nor does it typically yield data on what's been tested, defects reported to the ERP vendor, or quality trend. The implementation consultant has higher priorities, plus user acceptance testing comes at the end when *time-to-go-live* is fully compressed.
- **Hosting Provider.** Large companies typically look to an external provider to host and manage the application infrastructure – network, servers, databases, storage, redundancy, application technology stack – and obtain Service Level Agreements (SLAs) against which to manage the provider. The hoster has experience with the ERP package, and leverages lessons learned to design the system to address company locations, user volumes, database sizes, development and test instances, failover needs. But getting all these components working as expected – handling peak transaction volumes with acceptable response and measuring the fail-over impact of transactions in process – is an undertaking that requires testing under load. The hoster rarely has the capability to do this.
- **Management.** You've secured executive sponsorship, and likely committed to an aggressive timeline. You've established an executive steering committee, and you meet with them regularly, reporting on implementation milestones. Quality is implicit in your launch plan, and will be a gating factor for go-live. You'll want to provide management the metrics to enable them to base decisions on quality.
- **Business and Technical Teams.** All your teams contribute to some aspect of quality: business functionality, data, external interfaces, custom reports and

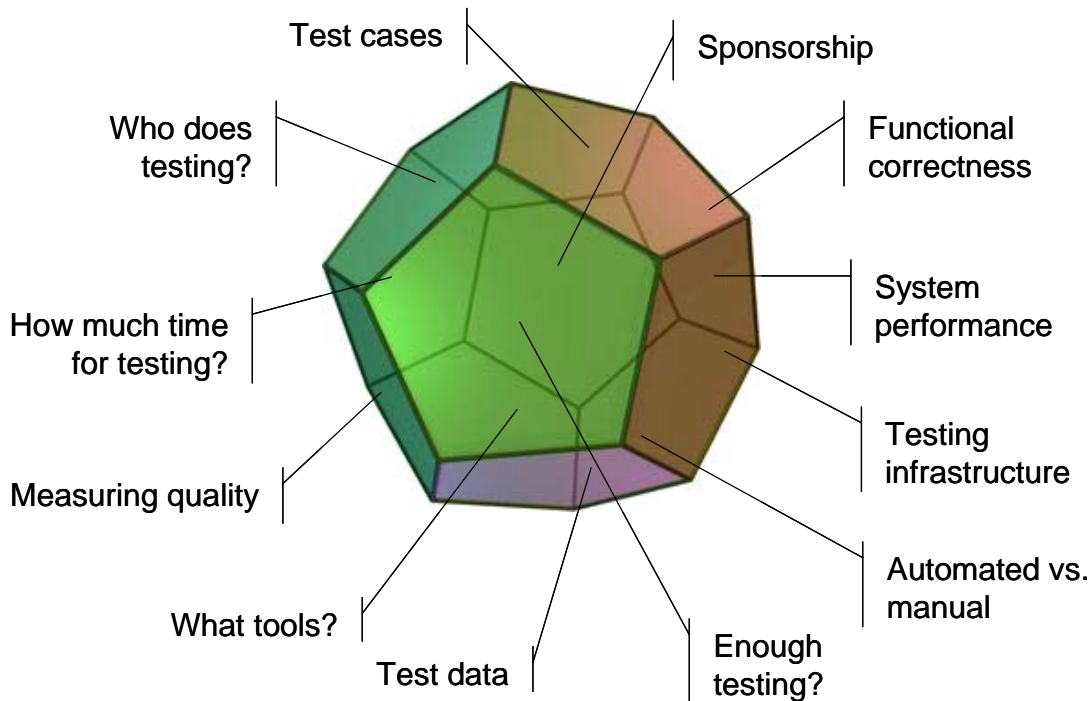
infrastructure. You'll want to make them accountable for it – not just at the end of the project, but from the beginning. You'll want to measure the right things, and make the results visible from early on.

- **Test Team.** You've taken to heart that testing is a separate checks-and-balances activity. Now, you'll want to plan and staff the activity properly. But you can't rely on internal application QA models for this. You'll need to plan training for your QA team to learn the application, alongside your business team. You'll want to plan sufficient time for QA teams to define, and Business teams to vet, test cases. You need test automation expertise to define and implement an automation strategy that will enable more, faster testing cycles, and that will pay dividends after go-live, through the endless cycle of patches and upgrades.

12 Quality Risks

Given the unique challenges and multiple competing stakeholder environment discussed earlier, if we now drill down to understand the implications on quality, we find a multi-faceted set of challenges that I have categorized into 12 areas.

These areas are depicted diagrammatically below.



Many of these areas should resonate with anyone experienced in software testing, and they set the stage for pitfalls to avoid.

Let's now drill down into recognizing and avoiding 12 mistakes that arise in managing quality in the context of unique ERP challenges and competing stakeholders and priorities.

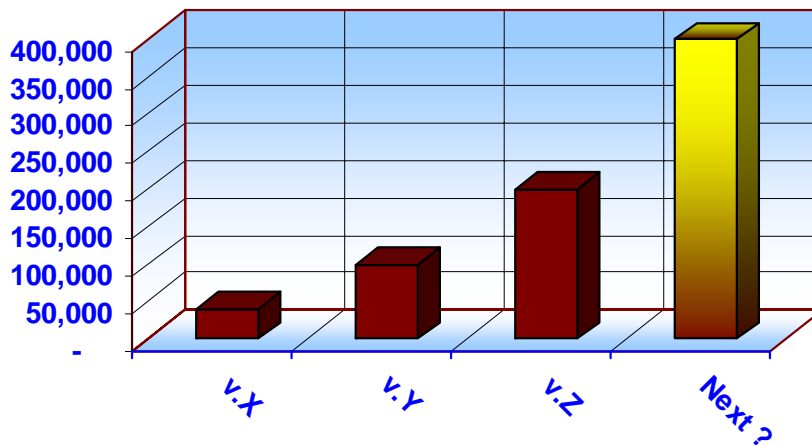
1 – Confusing product quality with implementation quality

Tell-tale Signs

While ERP software vendors like Oracle and SAP are touting great strides in release quality, industry analysts that interview customers and track quality and security issues over time remain bearish on the subject.

Without pointing fingers at any specific vendor, the defect trend chart below published by the Yankee Group in 2005 underscores their continued – and growing – concerns.

One Vendor's Defect Trend



Beyond those sobering statistics, one of the advantages of ERP software is their configurability to match your business processes. A major implementation activity is aptly known as “configuration”, wherein expert business users, after participating in product training, and make a myriad of decisions on user roles, workflows, business rules, and required and custom fields.

Risk

There is an underbelly to this flexibility – the software a company eventually deploys is a unique configuration of hundreds of choices that affect the software’s quality—and these must be tested, irrespective of the quality of the package. Failure to rigorously test this unique configuration risks incorrect or unexpected results and an unknown release quality.

Best Practices

To mitigate this risk, I recommend that you rigorously test both the end-to-end business scenarios – and their component functions – with the company’s production data.

And if your implementation is an upgrade, you should run the same scenarios on both the current release and the upgrade release. This will ensure that configuration choices made in the prior release are not adversely impacted by changed or new functionality in the upgrade.

Resources

Conducting good testing will require:

- A testing team
- An inventory of complete test cases
- Test automation and management tools

These will be elaborated as we discuss the remaining Mistakes.

2 – Trusting your hosting provider’s assertions

Tell-tale Signs

While there are niche hosting providers that specialize in specific ERP systems, many, especially the large providers, host a broad range of web applications and have no or low experience to draw on for designing the infrastructure to meet your capacity needs. They rely on the ERP vendor’s configuration recommendations or on the more limited experience of one or two individuals on their technical team, like a DBA.

Even when the hosting provider *is* experienced in your ERP software, they typically do not have the expertise to simulate and test your production load.

Risk

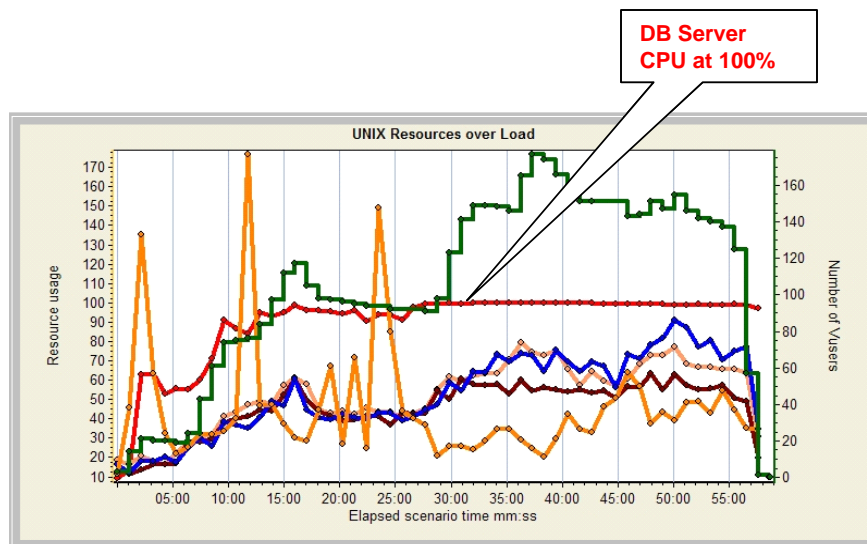
The risk is an under-configured – and under-performing – infrastructure that is not equipped to handle your peak transaction volume. Additionally, you will not be able to measure the true user impact of your site’s fail-over capabilities, as these will not have been tested under real load.

Best Practices

To mitigate this risk, there is no substitute for conducting a multi-user load test, simulating the peak workload that you expect the system to support.

Doing so can reveal component, applications and bandwidth bottlenecks in time for remediation before go-live. What could have been an initial poor user experience at best, or a productivity crisis at worst, can be averted.

System Resources Graph Illustrating CPU Saturation Under Load



Resources

Conducting performance testing will require:

- A cross-functional testing team, comprised of technical, business, and QA people

- A load testing tool that successfully supports the communication protocols and technology stack of your ERP product
- Performance testing expertise to plan and conduct a proper test.

3 – Failure to secure strong sponsorship for testing

Tell-tale Signs

If your company culture does not embrace Quality as a priority, or they pay it lip service without having process and people in charge of it, you may have a hard time securing executive commitment to invest and manage quality.

If you the Implementation Manager or Project Manager, another clue is that no separate plan and budget for testing—or it is “included” in the implementation consultant.

Risk

The risk of insufficient focus on quality should be evident – a bumpy go-live affecting selected departments at best, a major business process failure at worst.

Best Practices

As quality is implicit in every major implementation activity, to mitigate this risk ask each activity owner to identify, measure, and report on a Key Performance Indicator that relates to quality of that effort.

Remember the old saw *you can't improve what you can't measure*, and get activity owners to surface these measurements!

Resources

Gaining strong sponsorship can be helped by:

- Enrolling people in your company that *know* about quality, e.g., the VP of Development, your Director of QA, or the Manager of QC in Manufacturing, to help you develop a strategy
- Make quality measurements visible – as we will discuss shortly.

4 – Confusing *conference room pilot* with *test case development*

Tell-tale Signs

Perhaps the single most important ERP implementation activity is the *workflow configuration process*. While this activity goes under many names, on such name is *conference room pilot* – i.e., understanding the configuration options for each business process, selecting and entering these choices into the system, and then “piloting” the resulting custom implementation to validate that it supports your business functions.

As a key piece of input to this activity, process workflows are designed by the business team, and test cases follow, for the “piloting” step. These test cases usually identify menu navigations, data to enter, buttons to press, and results to expect – but are often documented loosely, with a lot of detailed knowledge assumed, given that the people creating these are experienced business users from each department.

If you are in charge of testing, and these test cases are what you are relying on, beware: they are usually incomplete, missing steps, don’t identify data dependencies as specific process flow into one another, and expected results may cover only what happens if the system works, not if there something is entered incorrectly.

Risk

The risk of inadequate test cases is a combination of frustration doing the more rigorous testing, insufficient time to test if test cases are fully defined first, or incomplete testing that results in uncertain quality.

Moreover, these test cases are focused on testing the business process, not the system load. Performance test cases are different in several respects: fewer, based on frequency of use and resource intensiveness, and quantified transaction volumes. Without these, load testing itself will be similarly impacted.

Best Practices

The way to mitigate this risk is to define a standard for a test case, with examples of what good ones look like – and make these someone’s deliverable. A couple of QA Analysts that take the conference room pilot test cases as an input would make good candidates for this.

A good (though simplified) test case template is an Excel sheet with these columns:

Process Name:

Step#	Navigation	Action	Data	Results	
				Expected	Actual

Resources

Get help developing robust test cases by:

- Using QA analysts to bullet-proof what the business team develops
- Put them in Excel, or even better, into a test management tools that supports entering (or importing) test requirements and manual test cases, and automated test script.

5 – Believing that testing can be done by the implementation team

Tell-tale Signs

To drive the implementation, you most likely you have engaged the ERP vendor's professional services team, or an implementation consultancy experienced with the package.

Undoubtedly, the implementation team has included testing into their proposal and work plan – though it is likely limited to *function* not *load*. Moreover, the implementation team's priority is configuring the system to support the business, and although they view testing as important, it is typically manual, business-user intensive, leverages incomplete test cases, and is not very repeatable.

If you are in charge of testing, and see no separate testing plan, budget, and resources, this is a likely sign too.

Risk

The risks of leaving testing to be done entirely by the implementation team are several:

1. Performance testing will likely not be done. This takes tools and special expertise that ERP consultants typically don't have.
2. You will likely have no/insufficient testing metrics to report on and gauge quality trends. The ERP team typically sees testing as a secondary priority, or a job that the customer should do; they are not expert testers, and don't have the experience or time to implement metrics.

The result will likely be insufficient test coverage, and weak quality metrics on which to abase management decisions.

Best Practices

Plan and separate testing subproject, and staff it according to implementation scope and test case estimates.

Resources

Get help planning effective testing:

- Your QA manager and ERP consultant to help estimate functional test effort
- Business users as go-to resources for your testers
- Tools for performance testing.

6 – Failure to allocate enough time for testing

Tell-tale Signs

If your project plan shows testing starting after configuration and development are complete, or if the same business users that are working on the configuration are also the ones doing the testing, these are sure sign of not enough time and resources allocated to this critical activity.

And of you *do* have QA people assigned to testing, keep in mind that with a packaged application your testers need time to learn the application before they can test it – and thus should participate in the same product training that business users took early in the process.

Risk

The risk of too little test time is obvious—incomplete or inadequate testing, resulting in potentially undiscovered defects in the release, a less than stellar go-live, and time-compressed, stressful fixes in production.

Best Practices

Start testing activities early—training testers in the application and establishing test environments (more on this shortly) for further exploration following product training.

Early on, develop a testing estimates based on hours per test case against which you can justify QA resource needs.

Pair up your best test analysts with business users experienced in the different application modules so they get to know each other and can leverage each other.

Resources

Get help planning and estimating test time:

- Use Excel to breakdown testing time and resources
- QA people—or outsourced testers – and business users
- An expert testing consultant knowledgeable in the application.

7 – Failure to make quality visible early-on

Tell-tale Signs

If quality is not reported on and discussed during leadership meetings, there are no or weak quality metrics for each phase of the implementation, or if you have no compelling graphs to summarize testing progress, coverage or defects, these are signs that quality is not fully visible.

Risk

You can't improve what you don't measure, and you can't enable management decisions on quality without metrics. This will result in an uninformed go-live decision with critical defects still open, or interfaces not fully tested. Again, a rocky go-live can be the result.

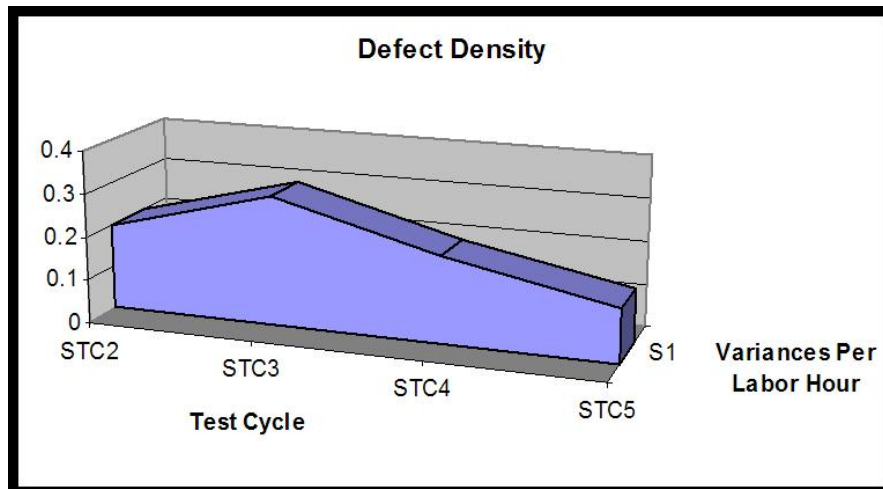
Best Practices

Tangible KPIs and intuitive visuals, coupled with some “management training” will make quality visible and an informed leadership team. Done early enough, this will also bolster weak quality sponsorship discussed earlier, resulting in more authority and budget.

Key metrics to report on fall into 3 categories: Progress, Coverage, and Defects. These three areas answer these questions:

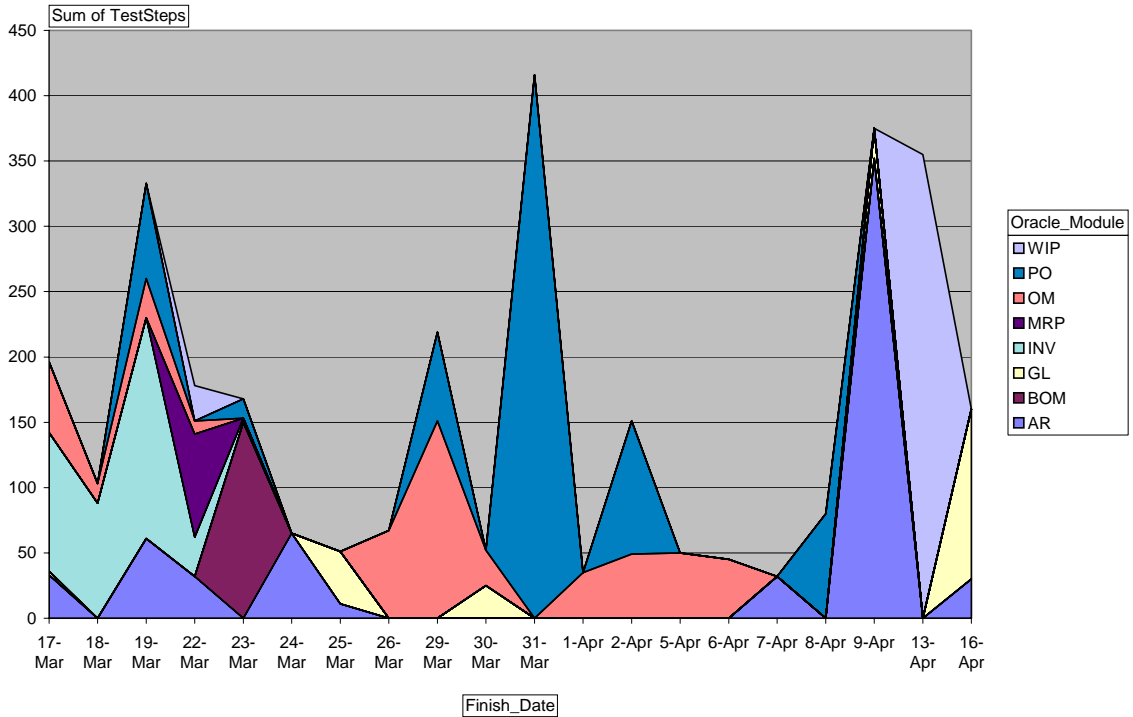
- How much testing have we completed? How much is left to test and when will we complete? Is our testing rate improving?
- What percentage of the application have we tested? How does this breakdown across application functions?
- How many defects have we found, fixed, or reported to the vendor? Are we fixing them faster than we are finding them? How long is it taking us to find them?

Below are two sample graphs from actual ERP projects. The first depicts a variation of defect density* across test cycles, and shows that after System Test Cycle 3 the test team is finding fewer defects per testing hour expended.



* Defect Density is strictly defined as “defects divided by complexity”. Here the definition has been modified to “Defects per Test Hour”

This second graph depicts planned test execution and coverage over time. Coverage is quantified in number of test steps (y axis), graph areas are color-coded to modules. The logical companion graph is to show progress against this plan – number of test steps executed, passed, deferred, etc.



Resources

Get help making quality visible:

- Leverage testing expertise to identify KPIs. Engage your implementation consultant or your QA manager in this discussion.
- Add start/end times to your test cases and have testers note how long they worked
- Use Excel to record the raw data, summarize (pivot tables), and graph
- Or use a Test Management tool with good reporting capabilities to automate much of the drudgery of using Excel.

8 – Failure to use the right tools to support testing

Tell-tale Signs

If your plan assumes all manual testing, you have no/low budget for test tools or a testing consultant, few defined testing cycles (typically you should have 3 to 6, depending on application scope), and your testing is all functional and not performance, you probably aren't properly tooled up to support a complete testing effort.

Risk

No tools to automate testing (how much to automate we will discuss later), ensures no repeatability, and no ability to test the infrastructure performance. Manual testing is resource-intensive and error-prone. After the first two times gets mind-numbing, and if you are relying on business users to test, they will start balking at your requests.

The result: inability to execute sufficient test cycles in the available time, no load testing, and again a potentially rocky go-live.

Best Practices

While QA often suffers from no tools budget, your best defense is an Excel spreadsheet with documented estimates with clearly identified and parameterized assumptions.

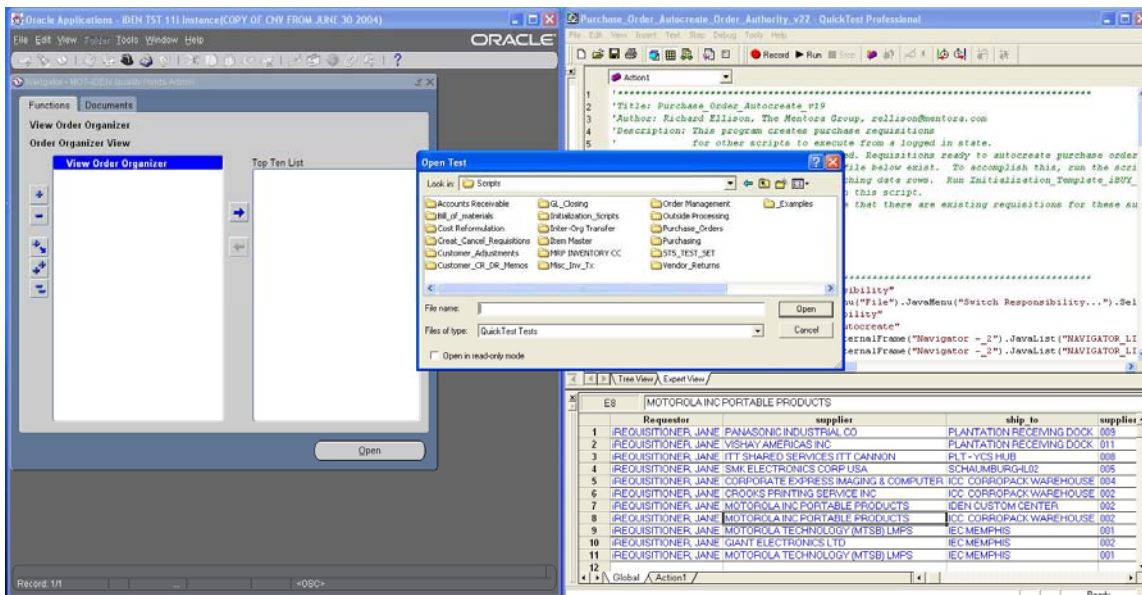
Show what you can do manual-only; then show what you could do with some automation. Do the cost-benefit analysis, leveraging vendor data and other case studies.

Resources

Power your testing and raise your testing efficiency with:

- Functional testing tools to begin building an inventory of automated tests. For some ERP systems, you can now buy script sets by module, and pay the vendor to configure then to your implementation

Screen shot of an automated functional test tool being used with Oracle 11i



Managing Quality in your ERP Project: 12 Mistakes to Avoid & Best Practices to Adopt

- Load testing tools to enable performance testing – there is no other alternative to this
- An expert testing consultant to help you sell the cost-benefit.

9 – Believing that automated testing replaces manual testing

Tell-tale Signs

We are all prone to ‘silver-bullet’ thinking – and more so when under pressure. The inexperienced manager may think that “automation means faster”, without stopping to consider that the initial investment required to implement automation itself takes time.

If your test strategy includes automation, but the project plan is the same duration – or less – than for all manual testing, this mistake looms.

The only area where automation *does* replace manual testing – e.g., by 20 people on a Saturday with lots of pizza – is in load testing.

Risk

The biggest risk is unmet expectations for automation, which may get the Quality owner a black eye, sour the organization on automated testing, and frustrate the people attempting to introduce automation.

Not to mention that you will likely get even less testing done than if it is totally manual, resulting again in incomplete testing and uncertain release quality.

Best Practices

Introduce automated testing selectively, on modules that have low/no customization, stable business rules, and no/few interfaces. Plan to automate 20% of test cases by test cycle 1, adding another 10-15% per test cycle. Plan cycles as a mix of automated and manual testing. Measure test cases executed and graph the trend that shows increased productivity.

Compute the cost of the initial investment – tools, training and script development, and subtract the cost of labor hours saved over each iteration – til you zero out the initial investment. Graph this, and use it to educate.

Resources

Plan a mixed manual and automated test strategy using:

- Functional testing tools
- Pre-defined scripts that can be purchased and configured to your workflows, if they are available for your ERP package
- Expert consultant to ramp up implementation and do knowledge transfer after go-live
- An expert testing consultant knowledgeable in the application.

10 – Failure to secure enough testing infrastructure

Tell-tale Signs

Identify the number and purpose of test environments that have been planned for the implementation. If there are not *at least two* dedicated to testing, dedicated tester workstations, and plenty of disk space for several copies of the production database, these are sure signs of insufficient testing infrastructure.

Risk

There is no quicker way to frustrate everyone involved with testing than to not have the computing resources needed– or to have to wait to use shared test environments. You will lose productivity, get less testing done than you’ve anticipated, and release quality will likely suffer.

Best Practices

Plan for at least these environments:

- One for Conference room pilot/Configuration, which could then become the environment for User Acceptance Testing
- Two for training – you’ll need at least two to train everyone in a timely fashion
- QA-1 for one set of modules
- QA-2 for another set of modules
- QA-3 if you are upgrading and need to run the same tests on the current version and on the new one
- Pre-production for load testing; if this is an upgrade, then you will need a separate production-like environment
- Plenty of disk space for test database as copies of production and for disk-to-disk refreshes (tape-to-disk typically take too long for enterprise size production databases)

Resources

Leverage these:

- Quality leader to push for these environments
- Infrastructure team to time to support the environments and database refreshes
- ERP and DB vendors to provide no-cost software for testing purposes
- Plenty of disk space for test database as copies of production.

11 – Underestimation of the effort needed to create reusable test data

Tell-tale Signs

If you have no plan for populating test data, creating master copies, and refreshing them within a few hours turnaround, these are environment flags.

If your test cases don't easily sequence in accordance with "macro business processes" like "Procure to Pay", "Order to Cash", or "Requisition to Purchase Order", and the data flow from upstream to downstream is not mapped and clearly understood, these are process flags.

Assess the sysadmin support allocated to the implementation. If the activity of refreshing test database is not identified, or takes longer than a few hours during the day or overnight, this is a support flag.

Risk

As soon as testing gets underway, the team will discover the test data challenge and will spin its wheels resolving this on the fly. Senior testers that know the business – and the data – well may get by, but more junior testers will be unproductive.

The net result will be low testing productivity and failure to meet the testing timeline.

Best Practices

Plan enough storage for disk-to-disk refreshes and 3-4 hour turn-around during the day, or overnight.

Obtain from the ERP consultant a copy of the process flows, understand them, and plan your test cases so they line up with them. Map the data entered or create upstream and identify which test cases need this data downstream.

Create test data for your test cases that yield a known result, and when done, get a master copy made so you can refresh from this master after a test full or partial test cycle is complete.

Plan sufficient disk capacity so that you can do disk-to-disk refreshes of test databases (tape-to-disk typically take too long for enterprise size production databases), and sysadmin commitment to turn these around promptly.

Resources

Leverage these:

- Test management tool for sequencing test execution and passing data forward
- ERP consultant for workflows
- Business users who know the company's data by domain for help mapping the data
- Disk storage.

12– Believing that you’re done with testing once you go live

Tell-tale Signs

In the heat of getting to go-live, it is common to think of it as a finish line. Once it is achieved, perspective shifts and it transforms into a *starting* line—for operational use and maintenance.

From a quality perspective, though, it is easy to believe that testing is truly complete and it’s time to move onto the next project. However, you discover very quickly that at go-live there are already patches from the ERP vendor that have been stacking up in the corner while you had frozen your go-live at a certain patch level. Then production use begins to reveal new defects undetected during pre go-live testing. Pretty soon you get to a point where it is necessary to apply at least selected patches.

The key question is: have you anticipate this moment, and planned your test strategy to include maintenance testing?

Risk

The risk is in installing patches with inadequate testing , or in having to hold back on installing them if testing resources are not available. Over time this can yield production defects not being fixed and/or not keeping up with business demands for leveraging enhanced functionality in a maintenance update.

Additionally, if this is a new release, there is likelihood that the environment will undergo changes—either in configuration or a rollout that will add a significant number of new users. If configuration changes are large enough, such as implementing a different database clustering approach or adding another middleware server, you will want to re-test for system scalability and capacity—or risk a performance crisis.

Best Practices

Plan for testing as an on-going process through the life of the application. Plan automation to play an increasing role in regression testing of stable functionality.

Plan for an additional performance test a month or so after go-live.

Resources

Leverage these:

- Automated functional testing tools and a growing inventory of robust regression scripts
- Level-of-effort QA resources for on-going testing
- A permanent test environment kept in sync with production, and a second where the patch can be applied and tested
- Your load test scripts and expertise to conduct an additional test.

Summary

ERP quality challenges are unique: by their scope, complexity, stakeholders, and competing goals. Quality is embedded in all key activities. Stakeholders will tend to put the best spin on the quality elements they bring to the table. Don't assume that the software, the infrastructure, the ERP consultant, or business users will address quality in a complete and disciplined way.

Make it a leadership topic from the get-go: Identify key KPIs, create a visual for each one that speaks in terms leadership understands and can act upon, and figure out what and how to measure to create generate these visuals for each phase of the project. Use these metrics to strengthen quality sponsorship.

Develop a test strategy, plan, and a budget that includes test automation – but don't fall prey to silver-bullet thinking. Leverage a little expert consulting to develop that strategy – someone that brings automation ROI experience and can help you set realistic goals.

If you don't have load testing expertise in-house, and your organization cannot justify purchasing load testing tools for a broader range of use, hire a performance expert and push the tools vendor to rent the tools for the performance testing period of the project.

Develop a comprehensive Excel template for documenting your test cases—or if your organization can justify it for a broader range of QA work, bring in a test management product in which you can create your test cases (or that you can import existing ones into), schedule and manage the testing effort, and track defects.

Remember that automation pays back over time – typically in 4 to 6 test cycles. Define plans and set expectations accordingly.

And remember: the life of any system – but especially of comprehensive ERP packages – is dynamic; and you'll face pressures to implement patches and upgrades that will require on-going testing. This is where automation truly pays off.

Watch for the tell-tale signs of these 12 common mistakes, log your own lessons learned, adapt the suggested strategies, and enroll the right people inside and outside your organization to help!

Good luck with your project!

About Mentora Group www.mentora.com

Mentora *tests, hosts and manages* business applications. We provide software quality and production management solutions across the application change lifecycle. We specialize in performance testing of web applications for eCommerce, eLearning, Donor Management, and packaged ERP suites from Oracle, SAP, and MACESS EXP. We host and manage business-critical applications in eCommerce, Healthcare, Insurance, and Publishing, at Savvis and AT&T tier 1 data centers. We have offices in Atlanta, Boston, and Washington DC, and deliver services nationwide.