Contact Center Technology Testing

Testing is essential when launching projects that introduce change into the contact center environment. How to ensure smooth transitions, upgrades and rollouts.

By Lori Bocklund and Matt Morey, Strategic Contact

Contact Center pipeline | **Pipeline Articles**
www.contactcenterpipeline.com

Lori Bocklund
Strategic Contact

Matt Morey
Strategic Contact

Imagine that you've finally secured budget to add new functionality and technology to your contact center. You've labored through vendor evaluations and inked a deal. You go full speed ahead on implementation to start living in the "promised land" of your newly enriched applications environment. *But wait*… as you cut over to production, the new stuff doesn't work quite right, and the old stuff is faltering. Before you know it, you're up to your eyeballs in trouble!

This scenario is all too common. The perceived "need for speed" puts pressure on internal and external resources to forego rigorous testing. The result? Downtime. Delays. Unplanned expenses. Workarounds that never get fixed and plague future efforts. Frustration for customers and the front line. And ultimately, degraded customer service. By contrast, thorough project testing allows for smooth transitions to new technologies and applications and upgrades to existing systems and services. Moreover, an overarching testing strategy includes testing processes to proactively prevent problems and speed problem resolution (see Figure 1, on page 3).

Project Testing

Testing is essential for contact centers of any size when launching projects (big or small) that introduce change to your environment. Project success depends on testing telephony and data infrastructure and applications, as well as applications integrated with the contact center, such as customer information systems or CRM. There are two primary testing tracks: functional testing and performance testing.

FUNCTIONAL TESTING

Functional testing occurs at various stages of the project cycle to ensure that the system delivers the features and functionality defined, and that, ultimately, users will succeed. Functionality testing includes:

- **Component testing** to ensure that specific elements or systems are working. Component testing is typically a vendor responsibility and is fairly narrow in its scope, but it is a key prerequisite for other testing.

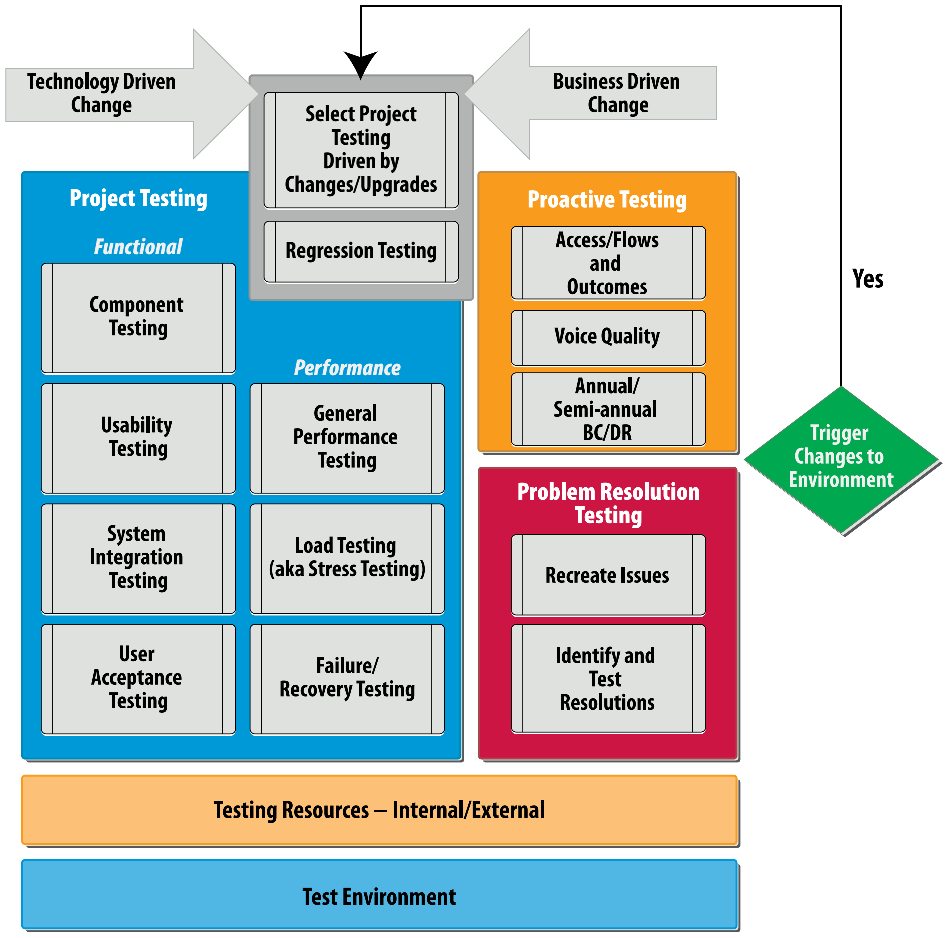
- **System integration testing or SIT** to ensure that all systems and applications work, and work together correctly. SIT should occur after the systems are installed and configuration or development is complete, and the vendor confirms that basic operation is successful through component testing. This form of testing addresses interoperability and call flows or workflows across systems. SIT also inherently includes "regression testing," verifying that any change to an established environment does not disrupt the proper functioning of existing elements. Vendors and IT have primary responsibility for SIT.

- **User acceptance testing or UAT** to ensure that the application functions according to business needs and requirements. UAT is typically conducted after the system configuration and development and core systems testing are complete. UAT is based on the requirements agreed to (and typically signed off on) by the users in the design stage and is primarily an end-user responsibility.

- **Usability testing** to assess the ease with which end users navigate the application. Usability testing goes a step further than UAT, confirming not just that the solution meets specifications, but that the users will succeed with what has been defined (and will be delivered). Usability testing can occur after design but before full development and integration testing is complete, using mock-ups and focus groups or other forms of feedback, such as video observation of the users as they interact with the system interfaces.

The Building Blocks of Successful Contact Center Technology Testing

Figure 1, *right*


Technology Driven Change
Select Project Testing Driven by Changes/Upgrades
Business Driven Change
Regression Testing
Project Testing
Functional
Component Testing
Usability Testing
System Integration Testing
User Acceptance Testing
Performance
General Performance Testing
Load Testing (aka Stress Testing)
Failure/ Recovery Testing
Proactive Testing
Access/Flows and Outcomes
Voice Quality
Annual/ Semi-annual BC/DR
Problem Resolution Testing
Recreate Issues
Identify and Test Resolutions
Yes
Trigger Changes to Environment
Testing Resources – Internal/External
Test Environment

These tests are highly detailed and should cover as many test cases as possible (e.g., IVR and routing paths, workflows, screens). Testing leaders need to work with IT/telecom, vendors and end users to create scenarios that emulate a broad range of actual transaction types. The most successful testing closely approximates the realities that will occur once the system is in production.

PERFORMANCE TESTING

Performance testing can take place at the beginning of a project to set a baseline, and near the end of the project (close to rollout) to prepare for production. It includes:

- **General performance** testing to assess application response times under "typical" load conditions. These tests may look at things such as screen pop or refresh times, IVR retrieval and message response delays, or report generation timing.

- **Load testing (aka stress testing)** to see if the application and infrastructure tolerate high volumes and capacity expectations can be met. Such tests cover network performance, as well as system(s) and interfaces under peak load conditions. They also reveal what happens when capacities are reached.

- **Fail-over for business continuity/disaster recovery (BC/DR)** to verify continuity in service despite removal of various components (e.g., servers, databases, gateways, network connections) and appropriate notification to IT or vendor support. These tests can also verify processes and timing for putting backup scenarios in place, and restoring normal configurations and operations.

As with functional testing, performance testing plans need to identify a full range of elements that should be put through their paces. You clearly don't want to unearth flaws in your system architecture, application design or processes when your technology gets slammed with traffic spikes or when facing your first outage.

Proactive Testing

Best-in-class organizations conduct performance tests on a routine basis to ensure that their technology continues to function at peak efficiency. Typical test plans exercise the IVR and routing algorithms while simultaneously assessing performance on inbound and outbound trunks. Here are some examples of proactive testing:

- Test calls to each dial-in number (or emails to each email address) ensure that all key contact paths are working effectively. These tests may occur as often as daily, and can ensure that the company knows of issues before a customer does.

- Testing tools assess the voice quality through the network to the agent desktops. In a voice over IP environment with a highly dynamic network, configuration or load, or with a history of performance issues, this type of proactive testing can be critical to optimizing the customer experience.

The Level of Effort Required of Project Resources Varies with Test Type, Table 1, *below*

	Typical Level of Effort for Testing		
Type of Test	**End-Users**	**IT/Telecom**	**Vendor**
Component—Test specific elements	-	LOW-MEDIUM	**HIGH**
SIT—Test applications work together	LOW	**HIGH**	MEDIUM
UAT—Test the features/functionality	**HIGH**	MEDIUM	LOW
Usability—Test the ability to navigate applications	**HIGH**	MEDIUM	SOW Dependent
General Performance—Test performance under "typical" conditions	LOW	**HIGH**	SOW Dependent
Load—Test performance and capacities under peak load conditions	LOW	**HIGH**	SOW Dependent
Failover or BC/DR—Test ability for systems to fail over and recover	LOW	**HIGH**	SOW Dependent

- Simulations can navigate a variety of paths through prompts and/or self-service applications and conclude with either disconnects or requests for transfer to agents. Call transfers into the IVR can also be simulated.

Perform tests at various times of the day and night to see what occurs under various load conditions, as well. Proactive BC/DR testing should only be conducted after-hours or during low-traffic periods to minimize the risk of a customer service-affecting incident. BC/DR tests are typically done once or twice a year for the core voice and data platforms to validate that fail-over is working correctly. Routers and gateways should also be included in such tests.

Problem-Resolution Testing

When an issue occurs, everyone is motivated to restore service as quickly as possible. A quick fix or workaround may be the short-term answer. But it's prudent to chase down the root cause of a problem and seek a resolution which ensures that the problem won't reoccur. That's when problem-resolution testing comes into full swing.

If you or your vendor has a robust testing environment, you'll have the means to attempt to recreate the problem without affecting customer service. You can leverage test plans and use cases from the implementation process to help develop a plan of attack. You'll modify the plans based on input from the help desk and/or examination of system logs. Your vendor can help to determine if the problems traces back to their code or your specific use of the application, or rule out that it isn't in those systems so you have to look at the network or other elements. Once you find the resolution, document the root cause and prepare a corrective action plan.

The "fix" from problem-resolution testing or the learnings from proactive testing may turn out to require a material upgrade or change to your system or application. If that is the case, you'll need to treat it as a project—even if it's a small one—and go through the appropriate protocols outlined earlier in the project testing section, along with regression testing. Similarly, business changes (e.g., new product, number, routing path, menus) or technology changes (e.g., upgrade software or hardware) trigger the need for some subset of these types of tests.

The Test Environment

Now let's look at the ideal environment for testing. Here's our short list of key requirements:

- Define the infrastructure elements and contact center applications that are essential to testing and therefore belong in a test environment. Any application that is customer-facing or impacting and/or you can't serve a customer without needs to be included. Beyond that, see what your budget can bear. IVR and routing, for example, are essential. You probably don't have to have workforce management or quality monitoring, but if you can find the funds, target the optional items that are most critical and/or most likely to present challenges in your environment.

- Build an environment that matches production as closely as possible. While it won't have the same capacity, it can have comparable elements, configuration and functionality. That means similar desktops (PCs, phones, headsets), integration and flows. You'll need to secure a small number of licenses from your vendors. You'll also need a test database, which may also double as a training database, to enable you to run realistic scenarios for inquiries and transactions.

- Procure hardware that mirrors your redundant and high-availability architectures. Include switches, gateways and routers. Similarly, build out a voice and data network that emulates the production environment.

- Consider a traffic generator for load testing and running scripts to test the IVR and routing to the agent desktop. You can purchase this capability or subscribe to a service. (See the Testing Resources section that follows.)

Of course, the challenge with many organizations is finding the funding to build out a robust test environment and staff to tend to it. The good news is most vendors sell test environments at a much reduced cost from a production environment. If your leadership recognizes the importance, plan to build multiple environments: one for application development, one to run test cases, one for quality assurance. The more changes you make (infrastructure or applications) or the more upgrades your architecture requires, the more important these environments become. If you lack a production training environment, one of these other environments can double as a training environment, adding to the importance and value.

For new implementations, establish a "live" test environment to complete thorough testing prior to cutover. For ongoing testing, if you cannot fund a testing environment, consider the following testing options:

- Work with your vendor(s) to facilitate lab tests for system integration testing (SIT) and user acceptance testing (UAT) on top-priority or high-risk changes.

- Run tests on a production system, but with "dummy" numbers and data to verify general fitness level (e.g., feature/function, general performance) without straining production capabilities, preferably during off-hours.

- Conduct any capacity and/or peak-load testing off-hours to ensure that the technology meets performance criteria and handles the worst-case scenarios.

- Use a controlled environment (aka "pilot") of 10 to 20 agents across two to four weeks to confirm readiness of features/functions, user interfaces or other customer- or user-impacting changes prior to full rollout.

- Always ensure that changes or upgrades have a rollback option in the event of failure. You should be able to quickly "undo" something that isn't working in production.

If you run a 24/7 operation, the case for a test environment grows stronger. With no off-hours testing window, you either risk production impact, forego testing, or shut down at various times to conduct testing. None of those scenarios are ideal, so make the case for mitigating risks with a test environment.

Testing Resources

Resource planning for testing can prove challenging. As shown in Table 1 on page 4, you'll need to engage end users, IT/telecom and your vendors at varying levels of effort depending on the type of testing you'll perform.

As you create project plans, allocate sufficient resources and time to perform testing. You need detail-oriented people involved in testing, and test leaders who can build test plans and cases, processes and tools for capturing testing outcomes and ensuring follow up where needed. Plan to run multiple testing cycles, as you'll discover items that require fixes and then retesting.

If your vendor has responsibility for implementing new technology in a project, address the testing requirements when defining their Statement of Work (SOW)—and ideally when selecting a vendor. Keep in mind it may be a vendor partner (aka distributor) that has this responsibility. Make sure that they identify their roles and yours for each type of testing to avoid surprises. You may find that they do not do as much testing as you might expect.

Vendors/distributors can also bring in tools or partners to conduct specific types of testing, or you can secure such testing resources directly. For example, Empirix and IQ Services are two companies that are very active in delivering testing to contact centers. They typically address load or stress testing, as well as potentially SIT, general performance and UAT in projects, as well as ongoing proactive testing. Empirix provides the Hammer tool, purchased to run tests with simulated contacts. This type of tool is a fit if you are going to routinely conduct tests and dedicate the resources to use the tool and analyze and act on the results. This scenario typically occurs in larger companies or ones with a very dynamic environment and a commitment to minimizing risk. If you don't have or can't secure that sort of testing staff internally, it's better to rely on external services to conduct tests for you. IQ Services or Empirix will conduct tests and provide online access to results. These services can be essential for large project implementations, but can also support ongoing testing to make sure that your center is operating well and your customers will experience a successful contact flow. For example, IQ Services has "heartbeat testing" and "virtual customers" that can conduct routine checks on your infrastructure and applications by sending or receiving contacts (calls, emails, web, etc.) and ensuring that the results match expectations for messages, performance and outcomes.

The Payoff

Testing is not the most glamorous of disciplines. In fact, it can be downright tedious. And, admittedly, it is sometimes difficult to muster the resources and the organizational will to make it a priority. But as contact center technology gets increasingly complex with a growing number of things that can—and do—go wrong, you can't afford to ignore the imperative to do it. Systematic testing can get you to the "promised land" with less pain, expense and delay on projects and changes to your world. An environment, processes and resources for ongoing proactive and reactive testing can help you to maintain effective operations, preventing needless problems and resolving issues more quickly.

Lori Bocklund is Founder and President of Strategic Contact.

✉ lori@strategiccontact.com
☎ (503) 579-8560

Matt Morey is Lead Consultant at Strategic Contact.

✉ matt@strategiccontact.com
☎ (913) 681-5133

The Importance of Usability Testing

Usability testing is particularly important for applications such as interactive voice response (IVR) or other interfaces that impact customers, or major changes to agent desktops or other tools. In today's world, the customer interface considerations could include mobile and web applications integration into the center through interfaces such as webchat and collaboration. Unfortunately, usability testing often shows up at the 11th hour, if at all, when there is little room (or patience) for making changes. More often than not, staff learns to live with things they don't like, customers suffer, and/or the resulting project outcomes don't meet expectations.

Do yourself, your company, your front line and your customers a favor: Conduct usability tests early (and often, if appropriate) so that you can make the necessary modifications prior to production rollout.

Pipeline Publishing Group, Inc. PO Box 3467, Annapolis, MD 21403
(443) 909-6951 | info@contactcenterpipeline.com