SEVEN KEY TACTICS FOR ENSURING QUALITY
INTRODUCTION

Besides avoiding disasters and fatal flaws, quality assurance (QA) delivers significant benefits for banks. Strong QA planning provides the groundwork for effective and efficient testing, which ensures new solutions will meet – and even exceed – customer expectations. This e-book offers seven key tactics financial institutions can execute in order to improve quality planning within their product development initiatives.
In developing new banking products, QA covers the systematic process of checking whether the new solution addresses all the specified requirements. A comprehensive QA strategy provides the guidance and structure for those related activities. In the diagram below, a typical approach to a software development effort is depicted in the seven chevrons in the middle of the graphic. Development starts with a solution design working toward installation and then support of the completed new system.

The activities shown here include how joint reviews (JRs) serve as a forum to resolve any outstanding quality issues before the next development step can proceed. Verification of the application requirements and the application design is critical to establish and document mutual understandings – as is the case with high-quality initiatives. The QA strategy outlines the framework for when and how these key activities occur within a program or software development initiative.

### Establishing the QA Strategy

**QUALITY PLANNING**

**CONTROL OF DOCUMENTS**

**VERIFICATION**

- Solution Design
  - JR
  - Issues
- Application Requirements
  - JR
- Application Design
  - JR
- Build
  - Defects

**DEFECT MANAGEMENT**

**PROBLEM RESOLUTION**

- Approvals
  - JR (Joint Review Results)
- Verification Results
- Test Results
- Issues

**CONTROL OF QUALITY RECORDS**

**VALIDATION**

- Install
  - JR
- Support
  - JR
- JR

**VERIFICATION VALIDATION**

- JR (Joint Review Results) Approvals
- Test
- Defect Reports
Bankers often have a hard time distinguishing between test strategies and plans, but one can’t exist without the other. Plans are concrete, while strategy documents provide vision and governance. In the IT world, a test strategy defines the QA activities to be performed during the software life cycle. It describes the responsibilities and authorities for accomplishing software QA activities and identifies the required coordination of those activities with other activities within a project.

A test plan, on the other hand, describes the plans for quality testing of software applications and software systems. It describes the software test environment to be used for the testing, identifies the tests to be performed and provides schedules for test activities.

The following table provides an overview of some of the key distinguishing points between a test strategy and a test plan.

<table>
<thead>
<tr>
<th>TEST STRATEGY</th>
<th>TEST PLAN</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PURPOSE</strong></td>
<td></td>
</tr>
<tr>
<td>Defines governance and ownership assigned</td>
<td>Defines scope of testing including what functionality will be tested</td>
</tr>
<tr>
<td>Defines overall approach for assuring the solution</td>
<td></td>
</tr>
<tr>
<td><strong>TEST APPROACH</strong></td>
<td></td>
</tr>
<tr>
<td>Defines test approach for each phase within the strategy:</td>
<td>Covers when specific testing techniques will be employed, e.g., regression testing, data validation, test limits</td>
</tr>
<tr>
<td>system integration testing (SIT), user acceptance testing (UAT), performance testing</td>
<td></td>
</tr>
<tr>
<td><strong>TIMING</strong></td>
<td></td>
</tr>
<tr>
<td>Defines entry, exit and acceptance criteria for test phases</td>
<td>Provides testing schedule and calendar to stage specific activities</td>
</tr>
<tr>
<td><strong>DELIVERABLES INCLUDE</strong></td>
<td></td>
</tr>
<tr>
<td>Test communications, reporting and metrics, allocation release plan, defect management and approach for change requests</td>
<td>Reference to test scripts, documentation of existing testing constraints, definition of defect levels and test criteria</td>
</tr>
<tr>
<td><strong>OTHER</strong></td>
<td></td>
</tr>
<tr>
<td>Defines testing terminology and testing objectives</td>
<td>Includes different types of test plans for integrated testing, application testing, systems testing, etc.</td>
</tr>
</tbody>
</table>
3. IMPLEMENT KEY QA ACTIVITIES

Beyond developing a test strategy and test plan, there are other QA activities to implement at the beginning of a program to create a foundation for success for the overall effort. They include:

<table>
<thead>
<tr>
<th>DEVELOP DOCUMENT CONTROL</th>
<th>ENSURE RECEIPT OF QUALITY RECORDS</th>
<th>REQUIRE JOINT REVIEWS TO FACILITATE DOCUMENT APPROVALS</th>
<th>EXPECT A QA SUMMARY REPORT</th>
</tr>
</thead>
<tbody>
<tr>
<td>A program and related supporting projects should generate multiple supporting documents. Thorough quality practices require complete document approval and sound recordkeeping. Program stakeholders need to know that only one version of the truth exists, and they need to know where to access that critical document.</td>
<td>Evidence that a quality activity was completed must be logged and uploaded into a readily available document repository. These records include approvals, verification of results and actual test results.</td>
<td>The joint review process is also established during the planning phase of the initiative and is critical to QA governance.</td>
<td>Before the client or program sponsor begins their phase of solution testing, they should review the findings in a QA summary report. This report details the defects and resolutions their partner uncovered and addressed before transitioning testing activities to the client or sponsor.</td>
</tr>
</tbody>
</table>
Verifying the requirements of the new solution is another critical activity within the QA process. The QA team must first identify deliverables to be verified and then define quality standards for each deliverable type. Next, they must review, approve and release verification standards for use in the initiative. Authors of any quality documents should produce deliverables to meet these standards.

The QA team should log problems when quality standards are not met. Problems achieve resolution when a deliverable meets quality standards. After confirming the deliverable meets quality standards, the team releases it for use by the program. A partial example of the classification and logging of defect codes for one recent FIS-managed QA program is shown below.

<table>
<thead>
<tr>
<th>DEFECT CODE</th>
<th>CATEGORY</th>
<th>DEFECT DESCRIPTION</th>
<th>IMPACT</th>
<th>EXAMPLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>BR-01</td>
<td>Not complete</td>
<td>Terms and units of measure are not defined.</td>
<td>Multiple interpretations of solution; incorrect design is developed.</td>
<td>The acronym CST is used as a requirement, but it has not been defined.</td>
</tr>
<tr>
<td>BR-02</td>
<td>Not traceable</td>
<td>A business requirement is not uniquely identifiable.</td>
<td>Inability to trace requirements forward or backwards</td>
<td>Each requirement needs to be numbered, e.g., 3.1.3 Requirement Description.</td>
</tr>
<tr>
<td>BR-03</td>
<td>Not traceable</td>
<td>A business requirement cannot be traced back to a stated business scope item (i.e., the scope of the solution has “crept” beyond the business need/objective).</td>
<td>Scope Creep – Solution delivers more than stated business need/objective.</td>
<td>Refer to a requirements traceability matrix.</td>
</tr>
<tr>
<td>BR-08</td>
<td>Not relevant</td>
<td>The requirement is a solution.</td>
<td>Design is constrained by the requirement.</td>
<td>Requirement is stated as: BeB will call a standard Connectware Deposit Account Balance inquire service to verify the funds in the original account.</td>
</tr>
<tr>
<td>BR-11</td>
<td>Not traceable</td>
<td>The requirement cannot be checked for correctness by a person or a tool.</td>
<td>It is not possible to confirm that the requirement has been met or, conversely, not met.</td>
<td>Example of untestable requirements: • System performance shall run fast. • The system shall be easy to use.</td>
</tr>
</tbody>
</table>
One technique to ensure full test coverage of a client’s requirements is the use of a Requirements Traceability Index. A solution architect creates a solution design that captures client requirements. QA experts then use the Requirements Traceability Matrix to ensure solution design requirements align with product definition documents created by the various product teams involved in the creation of the client’s new solution. This index ensures the allocation of each bank requirement back to the product requirements. This entire matching process helps achieve a high degree of confidence that all bank requirements receive the optimal testing coverage.

5. UTILIZE A REQUIREMENTS TRACEABILITY MATRIX

Requirements Traceability Matrix

Ensures client requirements have been fully allocated throughout the project

AT THE APPLICATION LEVEL

Ensures full test coverage of client requirements
Besides carefully tracing and testing the bank’s requirements and maintaining solid QA documentation, other quality techniques can add significant value to banking initiatives. Failure mode and effects analysis (FMEA) is one of them.

FMEA started in the 1940s in the United States as a way to improve manufacturing for the war effort. Although it began 70 years ago, this process remains valuable to the banking industry today. Because financial institutions compete globally against more nimble nontraditional financial services providers, they need to develop products quickly and with the high quality that encourages customer loyalty.

In times of scarce QA and technical resources, testing priorities must be determined in order to best mitigate project risk. FMEA lays out a clear and straightforward process for performing that critical ranking. The design elements or components that compose a complex solution are analyzed by functional lines of business in order to:

• Identify the potential failures that could impact the customer experience
• Identify the potential causes of each failure
• Assess the resulting impact on customer experience should the failure occur
• Determine the probability that the cause will occur and subsequently result in the failure
• Identify the current controls or the ability to detect the failure or cause

The FMEA focuses QA and testing resources in development efforts. Few banks have the luxury of unlimited time and technical resources. Prioritization in the testing can lead to fewer defects needing correcting at more expensive points in a product’s life cycle.

6. EMPLOY FAILURE MODE AND EFFECTS ANALYSIS AS NEEDED

In times of scarce QA and technical resources, testing priorities must be determined in order to best mitigate project risk.
Over time, the long-term impact of QA on a program needs to be evaluated and assessed in order to evolve and improve. A robust continuous improvement methodology should collect quality metrics from a wide variety of sources, analyze the data and contribute to the adjustment of QA processes. The methodology should also incorporate best practices from other complex initiatives and provide a forum for open discussion of lessons learned. These improvement activities can occur as milestones are reached in the program and during a postmortem discussion of the initiative.

The continual improvement methodology powers a repeating cycle of **Plan, Do, Check and Act (PDCA)** that benefits and evolves from the changes spurred by the methodology, such that each repetition of the PDCA cycle results in better QA than the last.