WHITE PAPER

LEVERAGING A TESTING METHODOLOGY TO ENSURE HIGH QUALITY BANKING APIs

Venkata Griddaluri

IBS Open APIs Manager of Quality Assurance

Why Quality Assurance is Needed

The need for high-quality projects and development efforts is especially evident in banking today. The rapid growth of Application Programming Interface (API) technology accelerates the need for thoughtful, comprehensive approaches to ensure the strictest quality standards are met. This is especially true in the world of rapid-paced open banking development efforts.

The costs of inferior quality are very high indeed. Evidence comes not only from the banking industry, but also from other industries having suffered severe consequences due to lack of stringent quality control and quality assurance procedures.

A black eye for Amazon

Software quality firm SQS, asked quality industry professionals to vote on the top software failure incidents of 2014.[1] The highlights of that list included Amazon's embarrassing technical glitch during the holiday shopping season that caused prices of thousands of items to reduce to just one penny, giving eagle-eyed customers a treat. Scores of small family-owned businesses were left lamenting the error and nursing heavy losses, with some warning they could enter the New Year facing closure.

An American Airlines fail

Another example of a system not delivering the intended results—indicated by this headline:

"**WHOOPS: American Airlines without pilots for Christmas after scheduling system glitch**"[2]

Suncorp Bank system upgrade causes money to disappear

Within the banking industry, examples of insufficient attention to quality abound. System upgrades, new software releases, and bug fixes can cause issues. All too familiar to other organizations, Suncorp Bank experienced a system upgrade issue reported by IT News:

"A problem with an upgrade to Suncorp Bank's systems over the weekend caused money to disappear from customer accounts and sent others into overdraft. The unspecified glitch occurred on Saturday night during an upgrade to Suncorp's core banking platform. Some customers reported being hundreds of dollars overdrawn, leaving them locked out of their accounts, while others complained that substantial amounts of cash had gone missing."[3]

While not always life-threatening, errors in banking software development can cause severe consequences for banks, their reputations, and their customers. This paper presents strategies banks can take to help ensure high-quality in the fast-paced development of banking APIs.

A Proven Quality Methodology

In addition to avoiding disasters and fatal errors, Quality Assurance delivers benefits for banks in the form of more effective and efficient processes that bring solutions to the marketplace that meet – and even exceed – customer expectations. These outcomes resonate in the world of API development.

[1] Continuity Central Archive, February 2015
[2] MSNBC, WHOOPS: American Airlines without pilots for Christmas, November 29, 2017
[3] IT News, Suncorp system upgrade causes cash to disappear, February 20, 2017

A quality methodology provides a foundation for success

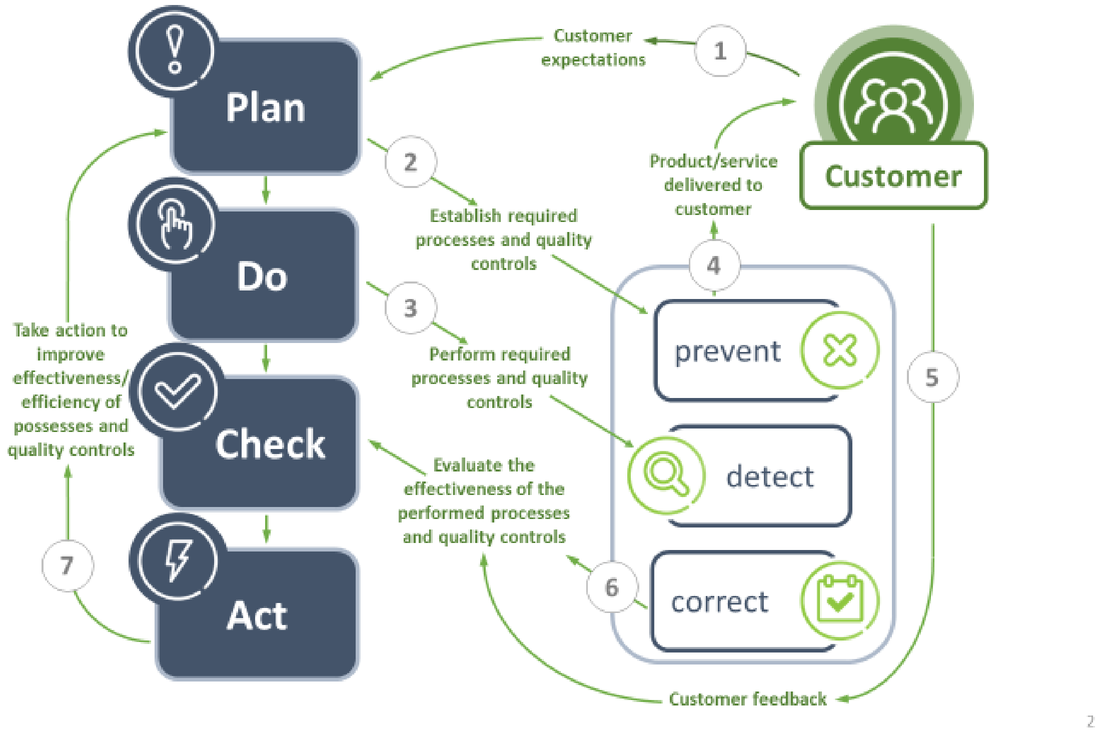
Using any of the various quality industry standards (such as ISO 9001 – Quality Management System Requirements, or the Software Engineering Institute's Capability Maturity Model Integration, CMMi) is an excellent way to establish a best practice Quality Assurance framework in your organization. These standards require your institution to establish quality controls throughout its business processes.

The intent of these quality controls is to ensure your customer requirements are met ***before*** your products and services are delivered to the customer.

Quality controls at the management level are commonly grouped into four phases: Plan, Do, Check, and Act (PDCA):

- Plan:
  - The customer's expectations and requirements are determined.
  - The processes and quality controls needed for the organization to meet the customer's expectations are identified and documented.
- Do:
  - The established processes and quality controls are performed – Prevent, Detect and Correct, below – to ensure the customer's expectations are met.
    - Prevent – The processes and quality controls are implemented to prevent the occurrence of a defect.
    - Detect – The processes and quality controls are implemented to proactively try to identify a defect.
    - Correct – The processes and quality controls are implemented to fix and resolve each identified defect.
  - The product or service is delivered to the customer.
- Check:
  - Feedback from the customer is received.
  - The effectiveness of the processes and quality controls are evaluated, answering the question, "Are the customer's expectations being met?"
- Act:
  - Actions are taken to improve the effectiveness and efficiencies of the documented processes and quality controls to meet the customer's expectations.

The following diagram provides a high-level view of the PDCA cycle:


Plan
Do
Check
Act
Customer
Customer expectations
1
2
Establish required processes and quality controls
3
Perform required processes and quality controls
Evaluate the effectiveness of the performed processes and quality controls
Product/service delivered to customer
4
prevent
detect
correct
5
6
7
Customer feedback
Take action to improve effectiveness/ efficiency of possesses and quality controls
2

API growth increases importance of quality and testing

A sound quality methodology based on the Plan/Do/Check/Act principles forms the foundation of Quality Assurance and includes the testing that must occur in API software development and integration efforts. The need for sound end-to-end testing becomes especially acute in solutions developed with multiple APIs. These individual components may interact differently when placed together in a more comprehensive solution that must respond holistically.

The growth of the API testing market provides another metric on the investments that firms are making to ensure high-quality API solutions. The API testing market is predicted to grow 20% annually, from $384 million in 2016 to $1.1 billion by 2022.[4] Next, let's discuss the crucial roles testing and Quality Assurance play in ensuring banking APIs meet user expectations.

Applying Testing within APIs

In the current era of rapid integration, using REST (Representational State Transfer) APIs, ensures quality is key to sustainable success. APIs host the critical business logic, which accept input data either from a user or from an application channel in the form of a request. The API then responds back to an end user or another application for further processing. Testing banking APIs and ensuring quality becomes more challenging as the scope expands beyond functionality into:

- Authentication

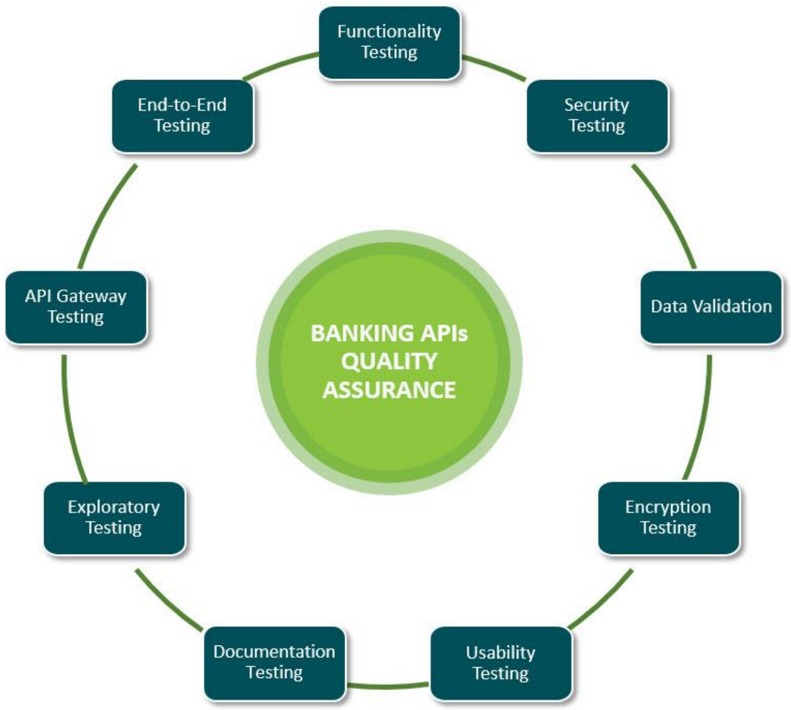
[4] Marketsandmarkets.com, *API Testing Market*, September 2017

- Authorization
- Data integrity
- Validating input parameters.
- Ensuring well-structured requests and responses suit the needs of the originating applications and database layers.

Testing areas of focus

Unlike traditional testing using the user interface layer, API testing focuses on the business service layer, which provides the tester an opportunity to explore negative and edge cases that may not be possible from the Graphical User Interface (GUI) tier.

Key Testing Areas for Banking APIs



Functionality Testing: This process performs various tests to ensure an API is working as functionally designed, but also is gracefully handling the failures by responding with desired status codes (see the HTTP status code table on page 6.)

Security Testing: Authentication and authorization are especially critical in banking APIs. Testers should ensure multi-factor authentication is performed prior to authorizing APIs to perform desired functions like GET, PUT, POST, or DELETE customer information.

Data Validation: In a banking ecosystem, several types of data can be accessed in an interface. This can include customer or account information, deposit data, loan information, transaction details, payment information, and real-time or end-of-day batch process details. Thorough validation should be performed on the input data, including:

- Data type validation
- Field length validation
- Data validation in the response body.

Encryption Testing: Because banking APIs deal with sensitive customer and account information, data encryption is especially important. As an industry best practice, encrypting authorization headers, URI path variables, or query parameters containing account or card numbers and any Nonpublic Personal Information (NPI) data elements is essential. FIS recommends verification and validation of the encryption information both at the API level and at the data field level.

Usability Testing: API testers should also ensure that APIs are designed and implemented for any developer to easily understand so that integrations can be built on them.

Exploratory Testing: The banking industry is currently exposed to many technology innovations such as Amazon's Alexa, Chatbots, and smart phone voice banking. Aspects of these new innovations should be considered and exercised in exploratory testing with reference applications or widgets that are integrated to banking APIs.

Documentation Testing: With rapid integration, financial institutions attempt to devise new products and services to cater to different and changing customer needs. Documentation provided with a new API becomes critical to providing basic knowledge about the solution. It should be reviewed for gaps and usability, as information shortfalls can impact the API's implementation and require additional support in the customer experience.

End-to-end Testing: Even though APIs are designed to support any originating application or a back-end system, performing the end-to-end testing using a presentation layer and a database layer helps to identify any data transformation or integration defects. It can help validate if APIs are presenting host returned messages in the response—which is important feedback to understand.

The testing previously mentioned should be performed during the integration and system testing phases. As testers move into regression testing, automation gains importance.

API Automation: APIs are developed and implemented at a rapid pace, both in an integrated environment and in silos. Building automation scripts for APIs helps reduce the overall testing effort and increases efficiency by reusing the same scripts across multiple environments by passing unique input parameters and test data. Remember to also schedule remote jobs that perform health checks of APIs and notification of any failures.

API Gateway Testing: An API marketplace presents APIs to end users so that they can subscribe, try out, and build integrations. When APIs are published in such a marketplace, prior to making the APIs generally available for end users, additional testing should be performed using the API gateway to ensure APIs are functioning as expected. Testing can be limited to a sample set of positive and negative scenarios.

API Testing Tools: Testers can take advantage of open source tools available for API testing. SOAP UI and Postman stand out due to their automation capabilities beyond the extensive functional testing competency.

The following is the general list of HTTP status codes commonly presented in API testing:[5]

Status Code	Message	Description
200	OK	Response to a successful REST API action. The HTTP method can be GET, POST, PUT or DELETE.

[5] owasp.org, *REST Security Cheat Sheet,* May 16, 2018

Status Code	Message	Description
201	Created	The request has been fulfilled and resource created. A URI for the created resource is returned in the Location Header.
202	Accepted	The request has been accepted for processing, but processing is not yet complete.
400	Bad Request	The request is malformed, such as message body format error.
401	Unauthorized	Wrong or no authentication ID/password provided.
403	Forbidden	It's used when the authentication succeeded, but authenticated user doesn't have permission to the request resource.
404	Not Found	When a non-existent resource is requested/Missing required data.
406	Unacceptable	The client presented a content type in the Accept header which is not supported by the server API.
405	Method Not Allowed	The error for an unexpected HTTP method. For example, the REST API is expecting HTTP GET, but HTTP PUT is used.
413	Payload too large	Use it to signal that the request size exceeded the given limit (e.g., regarding file uploads).
415	Unsupported Media Type	The requested content type is not supported by the REST service
429	Too Many Requests	The error is used when there may be a DOS attack detected or the request is rejected due to rate limiting.
500	General system error	The error is used as a generic response where the actual error may be available in the logs.

Best Practices in Banking API Quality Assurance and Testing

FIS recommends the following best practices for API testing that covers the functional and non-functional requirements within the development of APIs:

- Create a mapping of business requirements to APIs and prioritize the output so the priorities align.
- Verify the API is responding with correct status codes for all the positive scenarios.
- Verify the API is gracefully handling the invalid and negative scenarios and returning appropriate status codes.
- Understand the business flow and build scenarios with multiple APIs and test sequences and combinations.
- Focus on the data types and field lengths for all input parameters to ensure the API data validation is thorough.
- In the financial domain, encryption is critical. Test the API using more than one encryption method.
- Test the user guides and support documentation. Continually gather feedback from the developer community and improve the documentation going forward.
- Build automation for regression testing, verifying standard scenarios and any repeatable tests across multiple test environments.

Summary

A sound quality methodology provides the foundation for Quality Assurance in banking. A sound testing methodology further advances that foundation. This testing methodology should cover a variety of testing areas or functions, while also relying on proven best practices in software development in the banking industry.

As bankers look to collaborate more deeply with Open Banking partners, they should consider the quality and testing practices of those partners before any actual development begins.

Contact Us

For additional information on IBS Open APIs, and the quality practices supporting this type of financial solution, contact Amit.Aggarwal@fisglobal.com or call 414.815.1182.