

Electronic Payments Testing: A Testing Terminology Primer

Unit testing, integrated testing, functional testing, regression testing, system testing, acceptance testing—how do all of these terms relate to electronic payments testing? Everyone who uses your system is a tester of sorts, but who is responsible for each type of testing, and when does it occur in the development and testing schedule? This article provides an introduction to terms frequently used in financial transaction testing and describes the roles of the testers that are most often responsible for these processes.

To grasp any area of technology, one needs to be familiar with its unique terminology. Sounds simple enough but, in reality, the meanings of terms frequently vary from one industry to another—and sometimes from one company to another. Developers and testers of electronic payments technologies frequently use terms to categorize and classify the tests they perform, or to define their roles as testers. This article provides an introduction to the terminology used by electronic payments testers and provides examples of the various categories of tests.

Software Development Approaches: The Framework of Testing

Before introducing the terms that describe various types of testing, it might be helpful to look at a few examples of approaches to electronic payments software development, since these often determine the test approach favored by your organization. Your organization may use one of these approaches, or a combination of them, as determined by the needs of the project or preferences of your developers. (If your organization uses a different development methodology, investigate the testing approaches most often employed by that method.)

Agile Software Development

Agile software development uses face-to-face communication between developers, “customers,” and testers to develop software in fairly short-term development cycles. In the case of electronic payments software, a customer may be an internal customer, such as one of the system’s users inside the organization, or an external customer, such as a merchant acquirer. Because agile software development is collaborative, testers and developers work in tandem and Agile test plans and results may be spoken rather than heavily documented, as is typical of other approaches.

Waterfall Software Development

Waterfall testing is included in the more traditionally structured Waterfall software development paradigm. In the Waterfall model, software passes linearly and sequentially through several formally documented phases. Its sequential nature, with no iteration or overlap of any phase, is what distinguishes the Waterfall model from others. Typically, phases include requirements analysis, design, coding/implementation, testing, integration, and maintenance.

V-Model Software Development

Like the Waterfall model, the V-Model of software development is sequential, but it differs from the Waterfall model in two significant ways. The V-Model incorporates testing that is specific to each phase of the software development cycle; for example, code design would be followed by unit testing, integration design would be followed by integrated testing, system design would be followed by system testing, and so forth. Also, in the V-Model, each phase (and the testing that accompanies it) is iterative, so typically there will be multiple cycles of coding and testing.

Who is Responsible for Electronic Payments Testing?

In a sense, anyone who uses and evaluates software is a tester. Testers can be categorized into one of the following roles, each responsible for certain types of testing. Note that one person may serve in several testing capacities (that is, a QA tester may also act as an Acceptance tester), based on the available personnel and the structure and needs of the organization.

| | |
|---|---|
| Quality Assurance (QA) or Quality Control (QC) Testers | These testers may be organized as a QA or QC group or may exist as an extension of the development team. Typically, QA testers are employed specifically to test software and document the test results. They may have a professional certification in software testing or have an educational background in a related field. |
| Software Developers | These individuals are employed primarily to design or write code, and perform software testing as a means of verifying the operability of their code. |
| Business Analysts | These individuals, typically clients or system users, may not have strong programming or testing backgrounds, but will have in-depth knowledge of the business concerns that the software affects or addresses. |

| | |
|---------------------------|--|
| Acceptance Testers | These individuals may be clients or internal users who test an application with support from QA or development prior to delivery. If acceptance testers are not available, testing typically done by acceptance testers may be assigned to QA testers. |
| System Testers | These individuals are normally frequent system users, but may also be QA testers, business analysts, or acceptance testers. |

Classifying Tests: The “What” and “When” of Electronic Payments Testing

Previously, this article addressed the “who” of testing – that is, which roles in your organization may share the responsibility for testing during the software development cycle. The next step is to examine the types of testing that most often occur.

| Testing Class | What | When | Who |
|---------------------|---|---|---|
| Unit Testing | Testing the smallest piece (or <i>unit</i>) of isolated code before integrating it into the application. Often requires writing other supporting code (drivers and stubs) to conduct testing. Unit testing is designed to reveal coding problems early and thus reduce the cost of addressing the problem. | Any time there has been a change to or addition to the code, or when new functions are added. | Developers and, in special circumstances, (for example, when a module can be independently tested) QA Testers. If QA Testers are to assist in unit testing, the test plan should define what can be tested as a unit. |

| Testing Class | What | When | Who |
|---------------------------|---|--|---------------------------|
| Integrated Testing | Testing of multiple, <i>integrated</i> units. Integration testing reveals defects in the interfaces and interactions of the various units of code. Ideally, after unit testing is completed, the tested units are integrated and tested in various combinations until all modules in the process have been tested. Testing can be conducted top-down (integrating and testing the highest-level units first and thus testing high-level logic and data flow), bottom-up (integrating and testing the utility-level units first) or in an umbrella fashion (which uses some aspects of both top-down and bottom-up testing to test along data flow paths). | Any time there has been a change to or addition to the code. (Ideally after unit testing is completed.) Would be performed multiple times as each unit is incorporated. May also be coupled with regression testing for the previously integrated and tested units as new units are added for integrated testing. | Developers and QA testers |

| Testing Class | What | When | Who |
|---------------------------|---|---|--|
| Functional Testing | Testing the functional capabilities of the component or system from the user's perspective, <i>particularly as dictated by the functional design specifications and by industry standards</i> —for example, ensuring the program works as described in the design specifications and includes all the specified menu options. Functional testing is distinguished from Acceptance testing in that Functional testing objectively determines that the required functionality is included, while Acceptance testing typically involves the user's approval of the system. | Any time changes are made to add or modify a feature or system. Initially, some testing may be done using a prototype before the final coding changes are complete. | Developers (initially), but most often QA testers, Acceptance testers, and System testers (or other internal users). |
| Regression Testing | Re-testing pre-existing code to ensure defects have not been introduced by recent changes. The goal of regression testing is to successfully determine code that was previously tested and functional is still functioning properly after being combined with recently updated code. | Performed initially on existing code to create a baseline (to use for comparison), then performed again after the updated code has been integrated. | Primarily QA testers, but may also be conducted by Acceptance testers and System testers. |

| Testing Class | What | When | Who |
|-----------------------|--|---|---|
| System Testing | Testing the (integrated) software components with the associated hardware systems on which the software is installed. Some examples of system testing are communications testing, network testing, and stress or performance testing (such as testing system volume, system load capacity, or system uptime). Disaster recovery, failover testing, and backup/restore testing are also types of system testing. Notably, system testing can be required when there are no <i>code</i> changes; for example, when structural changes are made to a database, when a firewall is added, when routers are reconfigured, when hardware is upgraded, or when software is migrated to a new release. | Performed any time a change or addition is made to software, database structures, or systems or tools associated with the software. | Primarily, System testers, but may be conducted by others. For example, when a new firewall is added, an organization's security department would likely test the security of the system with the new firewall. |

| Testing Class | What | When | Who |
|---------------------------|--|--|---|
| Acceptance Testing | <p>As in Functional testing, this classification of testing is performed for the explicit purpose of verifying that the system satisfies the user’s requirements and meets the user’s needs. The focus of Acceptance testing, however, shifts to the user’s ability to successfully use the new features. Literally, Acceptance testing may be the final step before the user or another authorized entity “accepts” delivery of the system in acknowledgment that it performs satisfactorily. Acceptance testing can include Alpha and Beta testing (also called “field testing”). Note that the term Acceptance testing can also be used by some organizations to refer to smoke testing, or testing that ensures the code meets the established minimum requirements before turnover to another department. For example, an organization may require that before code can be transferred from Development to QA, a developer must have successfully installed the software on a PC. In this event, QA might perform an installation as a smoke test prior to formally accepting the system for testing.</p> | <p>At the point the system is thought to be ready for release, or any time there is a need to evaluate the system with regard to user requirements. In special circumstances, Acceptance testing may also refer to smoke testing that QA performs prior to accepting the system for testing.</p> | <p>Primarily, clients and Acceptance testers, but may also be conducted by Business analysts, System testers or QA.</p> |

Putting Test Classifications to Work During Electronic Payments Testing

This section presents two hypothetical situations: adding a new processing code (a change to your online, or *frontend*, processing) and adding a new report (a change to your batch, or *backend*, processing). Examples are provided to illustrate how each situation could benefit from tests in each testing classification.

| Situation | Examples of Testing From Each Test Classification |
|---------------------|---|
| Adding a new report | <p>Unit Testing: A developer uses a driver to feed report data into the reporting system, then a stub to create the new report. The output is extracted from the system and examined. The developer (or a QA tester) ensures that the report is formatted correctly, and that the report data is accurate.</p> <p>Integrated Testing: After unit testing is completed, the tester manually triggers each step in the process of generating and distributing the new report and examines the results. (For example, rather than adding the report to the automated report generator, the tester might use line commands to generate the report and to print it.) The tester verifies the ability to display the report online as well as examining the printed report for proper line feeds, page breaks, etc.</p> <p>Functional Testing: A tester examines the new report to ensure it satisfies both the user’s specifications and the organization’s standards; for example, examining headings, content, totals, spelling, line and page breaks, page layout, etc.</p> <p>Regression Testing: The tester ensures that generating the new report does not adversely affect existing report generation.</p> <p>System Testing: The tester verifies the setup for the new report’s distribution and that there is ample space for report storage.</p> <p>Acceptance Testing: After the new report has been tested and included in production, but prior to sending it to a client, an internal tester (such as a Business Analyst or an internal user) examines the report. Alternatively, developers or QA may work with a representative client to confirm that the report includes all the necessary data, is formatted as expected, and so forth.</p> |

| Situation | Examples of Testing From Each Test Classification |
|--|---|
| <p>Adding a new action for a processing code</p> | <p>Unit Testing: A developer sets up a test fixture that passes messages containing the new processing value directly to the module being tested, and implements a stubbed authentication module. (Note that regression testing could also be included as part of unit testing by passing known processing codes to the module being tested and validating that the module makes appropriate calls to the stubbed authentication module for each processing code.)</p> <p>Integrated Testing: A developer or tester uses a tool (such as a debugging tool) to manually or automatically send messages containing the new processing value to the processing system (Module A) after it is attached to a working authentication system (Module B). (Note that regression testing could also be included as part of integrated testing by validating responses [to existing processing codes] against expected values.)</p> <p>Functional Testing: The tester gives special attention to exercising the new processing code action and validating each step of its processing, including verifying any associated requirements for outgoing messages. All of the downstream processes, such as journaling, reporting, and posting should reflect the new action (as described in the user's requirements).</p> <p>Regression Testing: The tester must examine all existing processing code types and ensure there has been no processing change.</p> <p>System Testing: System testing may not be required for this situation, unless it is anticipated that transaction volume or system capacity might affect the proper processing of the action.</p> <p>Acceptance Testing: In the case of a new processing code established by a processor or network, the required Acceptance test may be certification testing with the network. Prior to implementation, network testers might set up an online testing session with a user to ensure the organization can accept and process the new values in the online message. The network testers and the client would discuss the transaction's processing path and results.</p> |

Your Next Step: Testing the Terminology

This primer has provided some basic information on electronic payments testing terminology. As indicated earlier in this article, however, your organization may use these terms differently. This primer provides a starting point for discussing these terms and tester roles with your peers so you can better determine the specific meanings established in your organization.

About Paragon Application Systems

Paragon Application Systems is a leading global provider of ePayment simulation, configuration and testing software tools to the financial industry. More than 400 financial institutions in over 80 countries use Paragon tools to improve quality and reduce time-to-market. Paragon's broad customer base includes major interchanges, processors, leading software providers, banks and credit unions. Visit Paragon Application Systems at www.paragonedge.com.

About the Author

Lynn Tjepkema, ePayment Specialist in Paragon's Professional Services department, is responsible for analyzing customer requests and developing working solutions to address testing needs. Lynn is ISTQB certified and has over 20 years of software development and testing experience in the financial services industry. She has extensive knowledge of financial network message protocols as well as authorization and settlement processing, with special emphasis on Visa, MasterCard and American Express.