Promotion of Tests

Once the assets have been persisted to a repository that all of your team's SOAtest users can access, we recommend defining a workflow based on the roles of the various team members.

Developers

In addition to unit testing frameworks such as JUnit and NUnit, developers require a functional test framework so they can stage the functionality of the system against a basic set of tests. In fact, such a practice is often required in agile development models and TDD (Test Driven Development) methodologies.

Developers' testing efforts can and should be used beyond the development group. Since developers are the most familiar with the technical aspects of the system under test, they are well-prepared to jumpstart the testing process by providing the initial building blocks for functional tests. For example, this might include defining building blocks such as:

- The initial tests for web services from a WSDL, WADL, OpenAPI/Swagger, or RAML definition—using basic parameter values that the service is expected to handle properly.
- The main use case flows through a Web site.
- The initially-configured database connectivity settings, with sample queries, for use in a SOAtest DB tool.
- Working JMS/JNDI settings.

Such initial test artifacts enable developers to provide QA a live and executable description of how to test the system—replacing and improving upon the "static" descriptions (such as Word documents with sample XML messages or wikis with instructions) that are commonly used otherwise.

As a result, Parasoft ideally recommends that developers have access to the same test projects as QA, and thus seed QA with test artifacts they can build upon. This enables QA to focus on the business requirements of the system: the end-user experience rather than the technical aspects such as connectivity settings, URLs, sample messages, etc. Typically, the tests produced by developers can be very basic and cover the main positive-path use cases. They tend to establish building blocks that indicate whether the system works, at least at a basic level.

Quality Assurance

In the recommended workflow, QA team members update their workspace with the sample tests created by developers and use that foundation to design the various use case scenarios around the system in order to discover what does not work. This may include:

- Extending test cases with data sources in order to increase test coverage.
- Designing negative scenarios (i.e., error condition test cases).
- Creating use case scenario tests to validate that the system adheres to all the defined end-user requirements.
- Designing end-to-end business process tests that may span multiple applications.
- Defining the test success criteria and validation assertions.

The tests created by QA should be committed to the shared repository so they are available to developers to execute as well. This enables developers to use the test cases to reproduce problems in the system, debug issues, and validate changes they make in the code as soon as they make them… without having to wait for QA to rerun the tests on the next drop.

Security Testing

Although many organizations tend to leave this critical practice as an audit exercise later in the lifecycle, most security experts—including Parasoft—recommend that security should be addressed as an inline process throughout the development lifecycle. The Parasoft solution framework follows that philosophy.

Parasoft SOAtest is used to augment existing functional test cases with security-related validations, which includes static analysis and penetration testing. The engineers who assume the responsibility of security validation (this may be the developers, QA or a separate group) can configure the security penetration tests on top of the functional tests. This is another reason why shared access to the test artifacts is needed and why collaboration on a shared workspace is extremely practical.

For full details, see Security Testing.

Load and Performance Testing

Some organizations have centralized load and performance groups who can access hardware and staging environments suitable for running realistic load tests. Others have the project teams perform this activity. Or, there may be mixture of both, with differences in scale, lifecycle stage, and frequency of tests. In all these cases (and regardless of who designs the load tests and who runs them to what extent), access to the functional testing project is extremely beneficial because performance tests are designed to run the existing functional tests under load. The tasks of performance testing engineers are:

- Retrieve the project (.tst) files that developers and testers added to the repository.
- Define load test scenarios using the existing unit and functional tests.
- Define the QoS (Expected Quality of Service) metrics to validate the services against SLAs and help identify bottlenecks using monitors.
- Commit the load test scenarios to the shared repository.

For full details, refer to the Load Test.