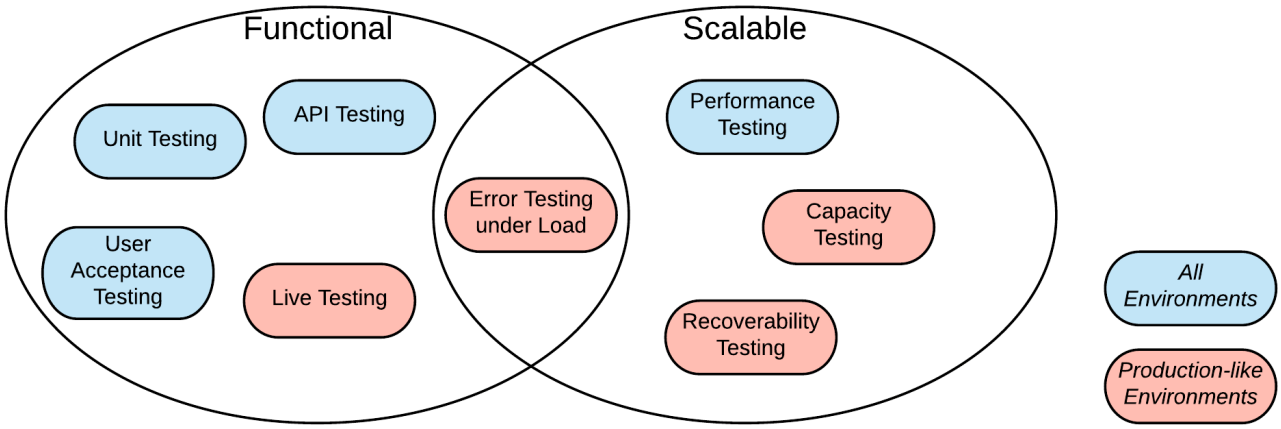
Testing Definitions

The goal of this document is to disambiguate different types of testing so that we can use the same vocabulary at JW Player. While the same process may be used to test different things, the goal of the test is what differentiates them.

Terms which are umbrellas of multiple types of tests are discouraged to avoid confusion, such as ~~Integration Testing, Load Testing~~ and ~~Stress Testing.~~



Chart

Scalability Tests

Performance Testing

Goal: Tuning systems as part of development process

Definition: Using varying levels of load to iteratively identify and eliminate bottlenecks. The goal is not finding bugs but improving performance.

Methodology: Define your expected load and the products performance SLA's. Given this, test by gradually increasing load while looking for bottlenecks. The workflow is “run load test > measure performance > tune system,” repeating until the performance meets your goals.

Capacity Testing

Goal: Find breaking point

Definition: Find out the maximum load it can handle while still meeting SLA’s. Find out how it will scale up or scale out - ie, what’s needed in terms of CPU, Memory or Network.

Methodology: Define your expected load and the products acceptable response time. Given this, test by gradually increasing load until SLA’s aren’t met.

Squeeze Testing

Goal: Identify bugs related to load

Definition: Exercise the system with the largest load it can handle, with the goal of exposing bugs which may otherwise be hidden (memory leaks, garbage collection, scaling issues, etc…).

Methodology: Put the system under the maximum expected load using a blend of queries, **for an hour**. The goal is to use a load which is near the limits, but not breaking the system. Monitor the system over an extended time, when all issues are resolved, performance should not degrade over time.

Recoverability Testing

Goal: Test how system recovers

Definition: Break the system in various ways, often by increasing load, or decreasing resources, in order to test its recoverability.

Methodology: List as many possible ways the system could be broken or overwhelmed as possible. From this list, choose the ones which give you the best coverage and execute them, making sure that recovery is automated.

Functionality Tests

Unit Testing

Goal: Does code do what we think it does

Definition: Code which automatically tests correctness of code after changes are made. They should run quickly and in any environment.

Methodology: Using unit_test for python testing, we should test the smallest composable bits of the system. This is generally public functions, classes and components. External systems are mocked to isolate the feature being tested.

Contract Testing

Goal: Validate API contracts

Definition: Does the API match the contract advertised by its documentation (i.e. its Swagger doc). Emphasis should be on finding edge cases and rare permutations that unit testing may miss.

Methodology: Automatically compare the return values of api calls against a contract specified by a Swagger document. This is done both manually and with Behave/Cucumber.

User Acceptance Testing

Goal: Align Product with Engineering

Definition: Periodic check-ins between internal stakeholders and the engineering team to manually verify that all groups are aligned on desired functionality.

Methodology: Access service as select strategic customers, run through some examples and keep an eye out for any assumptions in relation to business requirements.

Live Testing

Goal: Verify orchestration of services

Definition: Testing the full-stack by using real versions of all dependent systems in a production-like environment.

Methodology: Using real and synthetic user data to validate that pieces of the system can work together when queried from the outside. This is done both manually and with Behave/Cucumber. For example using Pingy to send ping data to the *real* ping nodes, which go to the *real* pipelines and to the *real* api's.

References

- Internal
  - MEP Load Testing Results - Capacity Testing in CICD
- External:
  - https://martinfowler.com/articles/microservice-testing/#agenda
  - http://agiletesting.blogspot.com/2005/02/performance-vs-load-vs-stress-testing.html
  - http://qastrategies.blogspot.com/2008/09/performance-testing-is-used-to-verify.html