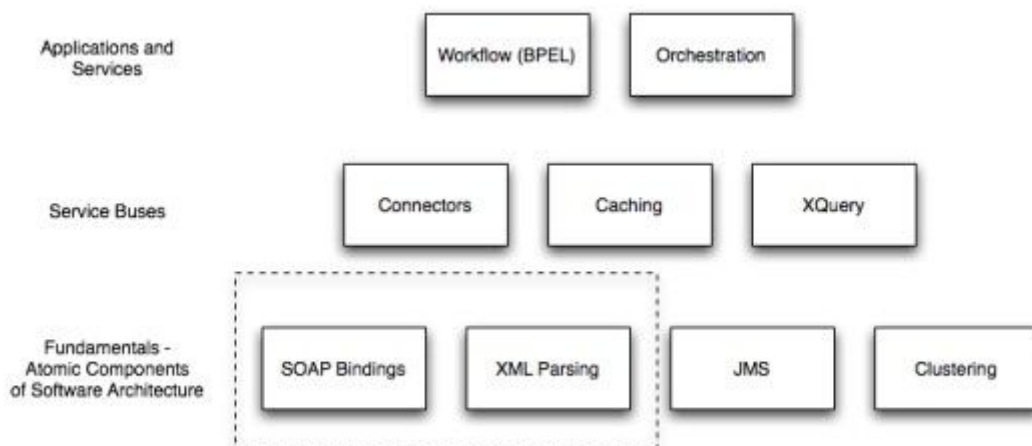


User Goal Oriented Testing (UGOT)

Countering software developers by saying "If you design for every possible user, no individual user will have their goals met when they use your software application!" Coverage tests are typically pointless, as users typically take a path through the functions in service, one after another, like chain links. The agile development community approaches this problem by recommending a test-first strategy, which is to write a unit test of a class before writing the class itself. At build-time, the compiler environment compiles the object code for the class and then runs the unit test against the compiled code. The unit test completes successfully by receiving example data and validating the response. If the class returns an invalid response, the unit test throws an exception that the build and deploy environment handles.

Unit testing and agile development methods help but are not a complete solution to UGOT techniques. For instance, a test first is usually only carried out on a unit level. SOA deploys applications as a collection of services, so testing individual units miss most of the big problems that occur during SOA integration and deployment. UGOT modeled tests check service as an individual user would – by picking one feature after the next in a chain of service requests, which is a more global and inclusive testing mechanism.

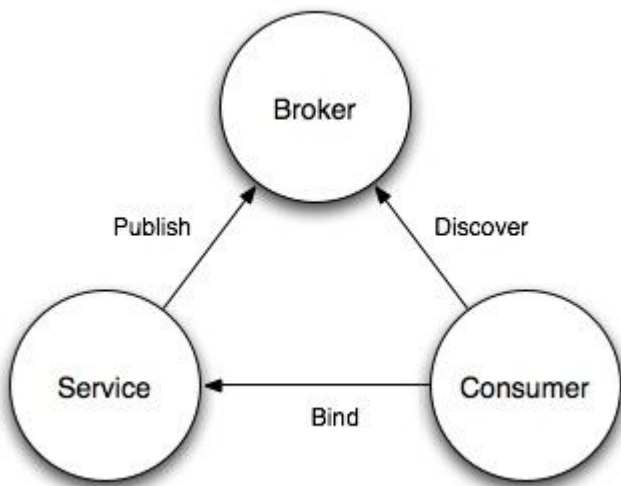
UGOT is ideal to understand performance and scalability. Internet applications must deliver real value to those who build and operate services. Understanding what is being tested is the heart of the issue, as can be seen in the following chart that shows the building blocks of Java development tools for building applications.



The components build on each other in three tiers:

- At the bottom are the fundamental components for soap bindings, XML parsing, JAVA inter-application messaging services (JMS), and clustering
- Building on these is service bus components for services to inter-operate at a message level
- The top tier provides interoperability at the application level

Given these building blocks and tiers, testing should be done across the tiers, where key performance bottlenecks in soap Bindings and JMS services may happen. Each of the components in the above chart impacts the scalability and performance of application implementation. Additionally, understanding the testing goal can be seen by reviewing the definition of service architecture as shown in the following illustration.



Which should be tested and when should it be tested? Performance tests normally check at least two. For instance, one test may check a consumer and service, and another check a consumer and a broker.

The Appvance Methodology lists options to understand what part of the system would benefit most from scalability and performance analysis.

Last updated: October 15, 2026

If you have any feedback on this topic, please contact us at documentation@appvance.ai

Applying UGOT

Define the user goal, the business outcome, and the sequence of actions that a representative user must complete. Use those goal-oriented flows to prioritize coverage and performance measurements, while retaining unit and component tests for lower-level defects.

