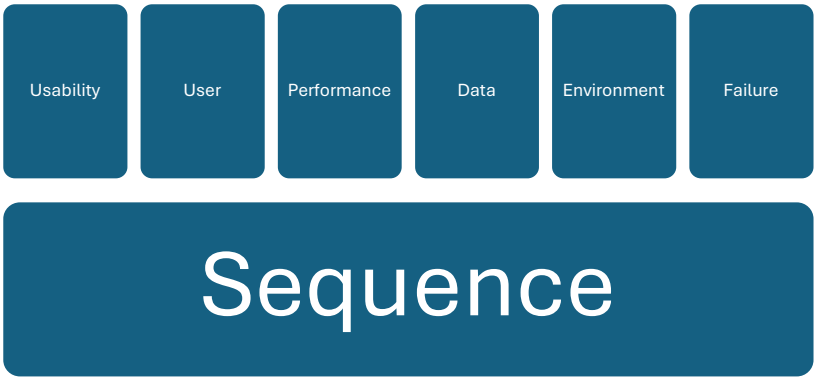
Heuristics for Flow and Scenario Testing

Flow and scenario testing are similar in that they focus testing activity and attention on sequence of actions and events, although for different purposes. Flow testing looks at sequence and order of events for its own sake to identify failures that may come of race conditions, concurrency, steps out of sequence, or unanticipated state, all independent of specific value of following those sequences. Scenario testing starts with an imagined purpose or intent from the user of the system, and constructs ways of using the product to satisfy that intent.



They are organized into the following categories:

Sequence: Sequence focuses on different ways of creating new test ideas and test conditions by causing steps and events to happen at different times and order.

Usability: Focuses on how easy the product is to use, and whether issues are present which impede ease of use.

User: Focuses on examining scenarios as needed or used by users in different roles and situations.

Data: Covering the scenarios in combination with different data values, states, and transformations.

Failure: Invoking and analyzing failure conditions mid-scenario to observe product impact and how the scenario is affected by the failure state.

Performance: Speed, throughput, capacity, scale and load implications of different scenarios on the system.

Environment: How aspects of and integration with different environments, environment states, settings and capabilities affect scenarios.

Sequence

Change the order of the scenario steps.

Leave steps out.

String different scenarios together one after the other.

Run the same scenario multiple times.

Abort the scenario and try to start over at the beginning with the same data.

Run a scenario concurrent with other system operations (backups, indexing rebuilds, schedule jobs).

Combine the steps of multiple scenarios together into a graph and exercise new paths that result.

Make longer scenarios out of single scenarios combined in different ways.

Create a state model diagram of product scenarios. Look for states that would be bad if they were ever to happen and figure out how the model can get there. Test by following different path traversals through the model.

Mix scenario steps with other actions or events that can happen in the system.

Test dependencies between actions and operations by looking for ways steps skipped, introduced or altered might fail to enable or block subsequent actions as expected, or might introduce conditions which will cause failure in subsequent steps.

Usability

Look through documentation for how to complete scenario and follow exactly as written without doing anything different. Note where instructions lead astray.

Identify critical workflows and trace their sequences step by step. Look for missing steps, undefined steps, or places where the next step might be a problem.

Look for confusing jargon, uncertain signals, difficult to understand or notice control states, or ambiguous meaning when working through the steps of a sequence.

Intentionally make mistakes while working with controls, data, and steps in a scenario and note any ways where the system fails or the user is unable to recover from the mistake.

Find someone who is not on the product team and ask them to try to accomplish a task. Do not give them instructions, do not give them hints. Watch them if you can. Take notes on their usage. Indicate any place where the user is unable to establish next steps or complete the scenario.

Instrument the product such that certain scenarios have markers in the diagnostic logs or telemetry streams with indications of scenario complete, error, or abort. Analyze the log to determine where users are having problems.

Compare the same scenarios against similar or competing products. Report instances where competition makes the scenario easier to discover, faster to complete, easier to control or understand.

Use the scenario via accessibility tools like screen readers or voice command.

Attempt to execute the scenario without using input devices you usually use – e.g. do not use mouse, use only keyboard, use only touch.

Change into high contrast mode, large display text mode and attempt to read screen.

Cover screen so you cannot see it and attempt to use it entirely without sight.

User

Run the scenario as users with different levels of access: lowest possible, highest possible.

Run the scenario in different user roles.

What role or person executes this scenario? Find out other tasks or work they do and build your own scenario to do those.

Attempt same scenario in a competing application and compare experience.

Combine your scenarios with scenarios for other applications people in that role are likely to use.

Construct sets of scenarios that represent role specific interests and workflows, sequencing them together if relevant, to simulate someone working in that role.

Attempt to execute multiple scenarios at the same time as different users.

Combine multi-user scenarios in ways that utilize similar resources or might expect different system state.

Combine users of different roles and authorization together on the same multi-user scenario.

Data

Change international settings for date, time, calendar, currency, keyboard, edit direction and attempt to execute the scenario.

Pair up the scenario with different data inputs that might prove challenging: upper ASCII, double byte, LTR versus RTL data entry, special characters, null and blank, improper data formats and types, extra-long inputs, inputs that violate stated restrictions.

Perform a data analysis of the system and combine scenarios with different data permutations.

Execute scenarios that cross storage systems and data processing components. Look for places where data is out of sync or not handled properly between multiple components.

Check scenario behavior with corrupted or invalid data and see how the product handles it. Look for lack of ways to repair, insufficient error messages, or extremely harmful outcomes.

Failure

Introduce delays or interruptions between sequential actions.

Execute a scenario and force a failing condition that interrupts mid-sequence.

Run a scenario while introducing operating system or environment failure states like starved resources, slow network connection, or low power shut down.

Scan system diagnostic logs and telemetry stream to see if any errors are emitted or anything appears wrong.

Use network traces, packet sniffers, or development tools to watch application behavior on network, file access or other behavior while executing scenario and see if anything looks odd, different, unexpected, or wrong.

Introduce intentional errors or invalid inputs at different points in the sequence.

Force a process to stop unexpectedly or fail midway to see if the system has trouble with graceful recovery or leaves other signs of permanent or problematic damage.

Force invocation of product error messages or dialogs during scenarios. Attempt to use the information presented to solve whatever problem is happening and complete the scenario.

Report any case where the end user ought to be able to correct the problem but is given insufficient or confusing information in the error.

Performance

Time scenario and see how long it takes.

Count the number of steps taken to complete a scenario and assess if that seems too long.

Run the scenario for a very large number of repetitions at high speed.

Perform IO and network traffic capture while executing scenario and watch for sequences of operations that might take longer, have gaps, or create loading dependencies.

Capture database and data storage fetch and write operations, query performance during scenario execution. Look for any patterns and behaviors that might lead to performance problems under load.

Analyze queries and other commands sent to sub-systems and storage tiers. Check against schema and architecture of stored data to see if queries and commands are appropriate, or if storage is configured to match use patterns.

Measure system utilization during scenario execution.

Load the system with large volumes of data and execute the scenario. Look for performance degradation when working with large data sets.

Scale the system to handle larger capacity and execute scenarios, paying attention to those whose resource use must span resources that cross scale units.

Environment

Run the same scenario across different devices, environments, platforms, browsers.

Start a scenario on one platform and switch to another somewhere in the middle.

Change environment settings (operating system, device, browser) to see how they affect the scenario.

Test how the scenario plays out in interaction with other systems (e.g., APIs, databases, or third-party software).

Simulate or invoke unexpected events or system behaviors while executing different steps in the scenario.

Execute scenario on a localized build.