

# Evaluation of Testing Limits LO4

s2298559

January 2026

## 1 Selected Requirement Deep-Dive: FR3 (No-Fly Zone Avoidance)

### 1.1 Requirement Summary

FR3 is a safety-critical requirement specifying that:

- The computed path must never intersect any no-fly zone polygon
- No-fly zones must be enclosed polygons
- Neither pickup nor destination may lie inside a no-fly zone
- If the pickup or destination is blocked off by zones, the system must refuse delivery

This requirement involves non-linear geometry, floating-point arithmetic, and global path constraints, making it ideal for assessing the limits of testing.

### 1.2 Implemented Test Evidence

FR3 is tested using multiple complementary techniques:

- Boundary and correctness tests (polygon closure, valid pickup/destination)
- Geometric invariant tests (no path point inside a zone)
- Property-based tests checking intersection of every path segment
- Adversarial configurations (donut-shaped zones, high-vertex polygons, nested zones)
- Scaling and stress tests with up to 40,000 synthetic no-fly zones
- Long-distance restaurants (Glasgow, Manchester, London) to challenge algorithmic scalability
- CSV-backed statistical performance testing with repeated runs and variance analysis

- Stubbed remote data to examine whether failures arise from dependency behaviour

Some evidence mapping can also be seen in Table 1

Aspect	Evidence
Polygon enclosure	<code>DeliveryPathCalculatorTest.testNoFlyZonesAreClosedPolygons_FR3()</code>
Pickup/destination not inside zone	<code>DeliveryPathCalculatorTest.testRestaurantAndDestinationNotInsideNoFlyZones_FR3()</code>
No path point inside zone	<code>DeliveryPathCalculatorTest.testPathPointsNeverInsideNoFlyZones_FR3()</code>
Segment intersection	<code>DeliveryPathCalculatorPropertyTests.testPathSegmentsNeverIntersectNoFlyZones_FR3.property()</code>
Blocked destination/pickup	Synthetic tests expecting "No path found to the goal"
External data validity	<code>ILPRestServiceIntegrationWithStubbedRemoteTests</code>

Table 1: FR3 evidence mapping to implemented test cases.

These tests cover structural, geometric, and behavioural aspects of FR3.

### 1.3 Coverage Gaps and Limitations

Despite this breadth, several important gaps remain:

- Infinite geometric input space: *No finite test suite can exhaustively cover all polygon shapes, orientations, concavities, self-intersections, or restaurant positions.*
- Precision-related edge cases: *Floating-point arithmetic may cause rare near-boundary failures that are unlikely to be discovered through example-based testing.*
- Path planner heuristic: *The planner may succeed in tested configurations but fail in others with similar geometry due to algorithmic heuristics rather than rule violations.*
- External data dependency: *Live ILP REST data may change over time; tests passing today do not guarantee correctness under future configurations.*

## 2 Statistical Reasoning and Confidence Limits

### 2.1 Use of Statistical / Property-Style Testing

Several tests use repeated or property-style checks, including:

- randomised GeoJSON path generation (50 samples)
- repeated invariant checks on generated paths
- testing all 16 permitted movement angles
- Randomised sampling of paths and polygons
- Property-based loop tests over all movement angles
- Repeated execution tests to measure variance in runtime and correctness
- Scaling experiments sweeping  $zones \in 1, 5, 10, 25, 50, 100, 200, 500, 1000, 5000, 10000, 40000 \times$   
*restaurant \in Edinburgh, Glasgow, Manchester, London* with 5 repeated runs each

Some of the tests produce a csv file and can be visualised statistically; some of the example insights include:

- London, with 40,000 no-fly zones has a runtime of around 70s
- Manchester, with 40,000 no-fly zones has a runtime of around 70s
- Glasgow, with 40,000 no-fly zones runs in around 36s
- Edinburgh, with 40,000 no-fly zones runs in well under 5s

And considering that the service is only made to support deliveries within Edinburgh, the service performs well over realistic expectations.

These tests improve confidence by sampling a wider input space than single examples.

## 2.2 Why Statistical Testing Is Still Limited

The absence of observed failures does not imply correctness. For example:

- Observing zero failures does not imply zero failure probability
- 50 or 100 runs only imply that failures appear rare under our sampling distribution
- Results depend heavily on the sampling strategy, not the true input space

If we model each test as a Bernoulli trial with unknown failure probability  $p$ , observing zero failures over 50 trials only implies that  $p$  is likely small, not that  $p=0$ . Thus, statistical testing improves confidence qualitatively, not definitively.

## 3 Coverage Analysis Beyond Code Coverage

### 3.1 Achieved Coverage Types

- Boundary coverage: pizza count limits, total price edges, angle bounds
- Error handling: malformed JSON, null requests, invalid polygons
- Geometric invariants: intersection prohibition, centroid constraints
- Behaviour coverage: path refusal under blocked scenarios
- Performance coverage: scaling behaviour across inputs
- Data coverage: real REST data + synthetic adversarial data

### 3.2 Missing or Weak Coverage

- Path optimality
- Extreme polygon complexity (high vertex counts, self-interactions), as tests don't explore pathological concavities, polygons with hundreds of vertices or fractal-like obstacles
- Mutation coverage, which would detect whether tests can catch small logic breaks but is not performed
- Concurrency / throughput such as testing 100 parallel deliveries for more realistic usage

## 4 Performance Testing Evaluation (QR1)

Performance testing includes:

- scaling tests
- statistical averaging
- distance-scaled experiments
- runtime variance measurement
- CSV-driven external analysis

There are some remaining limitations:

- Runs occur in a single runtime environment; results may vary significantly elsewhere
- The planner lacks a guaranteed worst-case bound, making performance probabilistic. Only single-request performance tested, not concurrency or throughput

Thus, performance testing gives high practical confidence but not formal guarantees.

## 5 Robustness and Error-Handling Limits (QR3)

Robustness testing shows:

- All HTTP endpoints correctly reject malformed JSON
- Direct-call controller methods can throw `NullPointerException` when validation annotations are bypassed
- Some synthetic restaurant injections initially failed due to real-data dependency
- Error propagation from `ILPRestService` depends on external API stability

These failing robustness tests are intentionally retained as evidence of limitations; they reflect realistic weaknesses.

## 6 Why Testing Alone Is Insufficient Here

- The system operates over continuous numeric domains realistically. Safety properties depend on global path behaviour, not local checks
- Correctness depends on external dynamic data
- Floating-point arithmetic introduces unavoidable approximation.

Therefore, even with extensive testing, full correctness cannot be established through testing alone.

## 7 What Would Be Needed for Higher Assurance

To approach stronger guarantees, the following would be required:

- Formal geometric reasoning or model checking for FR3/FR4.
- Stronger property-based testing with thousands of runs
- Mutation testing to detect specification sensitivity
- Load or performance modelling for profiling memory/time scaling under parallel execution.
- Stable offline world model with mocked REST data for reproducibility.