Week 5: Test Coverage Assessment Report

Date: January 31, 2026
Current Coverage: 9.49% (4,818 / 36,916 statements)
Target Coverage: 80%+
Tests Collected: 2,402

Executive Summary

The STUNIR project has comprehensive test infrastructure (91 test files, 2,402 tests) but low actual coverage due to many uncovered modules. Priority areas for improvement identified.

Critical Gaps (0% Coverage)

Core System Components (High Priority)

1. **ir/control_flow.py** - 810 lines - Core IR control flow analysis
2. **verify_build.py** - 731 lines - Build verification system
3. **ir_to_code.py** - 435 lines - Main code generation interface

Type System (Critical for Safety)

1. **tools/stunir_types/type_inference.py** - 502 lines
2. **tools/stunir_types/type_mapper.py** - 428 lines
3. **tools/stunir_types/type_system.py** - 427 lines

Semantic Analysis (Core Functionality)

1. **semantic/analyzer.py** - 438 lines
2. **semantic/checker.py** - 365 lines
3. **semantic/expression.py** - 501 lines

Emitters with 0% Coverage (26 Categories)

1. **business/cobol_emitter.py** - 595 lines
2. **beam/elixir_emitter.py** - 425 lines
3. **beam/erlang_emitter.py** - 341 lines
4. **planning/pddl_emitter.py** - 379 lines
5. **business/basic_emitter.py** - 338 lines
6. **asp/dlv_emitter.py** - 248 lines
7. **asp/clingo_emitter.py** - 232 lines
8. **constraints/minizinc_emitter.py** - 185 lines
9. **expert_systems/jess_emitter.py** - 147 lines
10. **memory/safety.py** - 287 lines
11. **memory/manager.py** - 182 lines

Low Coverage (<20%) - Needs Improvement

1. **systems/ada_emitter.py** - 8.7% (812 lines)

2. **scientific/fortran_emitter.py** - 11.2% (733 lines)
3. **scientific/pascal_emitter.py** - 12.0% (727 lines)
4. **functional/fsharp_emitter.py** - 9.1% (623 lines)
5. **systems/d_emitter.py** - 10.4% (539 lines)
6. **optimize/optimizer.py** - 19.9% (512 lines)
7. **codegen/cpp_generator.py** - 13.1% (472 lines)
8. **All Prolog emitters** - 15-19% (370-470 lines each)

Recommended Testing Strategy

Phase 1: Core System Tests (Target: +30% coverage)

- Type system modules (type_system, type_inference, type_mapper)
- IR control flow analysis
- Semantic analyzer and checker
- Build verification system

Phase 2: Emitter Unit Tests (Target: +25% coverage)

- Create template-based tests for all 26 emitter categories
- Each emitter should test:
- Basic code generation
- Type mapping
- Error handling
- Edge cases

Phase 3: Integration Tests (Target: +15% coverage)

- End-to-end pipeline tests for all 4 workflows
- Cross-language IR validation
- Real-world spec testing

Phase 4: Performance & Edge Cases (Target: +10% coverage)

- Memory safety tests
- Optimizer tests
- Resource management tests

Test Optimization Opportunities

Based on 2,402 tests taking significant time:
- Identify slow-running tests (>1s each)
- Parallelize independent test suites
- Mock expensive operations (file I/O, subprocess calls)
- Use test fixtures more effectively

Next Steps

1. Create comprehensive unit tests for type system (estimated +15% coverage)
2. Add emitter unit tests using templates (estimated +30% coverage)
3. Enhance integration tests (estimated +20% coverage)

4. Add edge case and safety tests (estimated +15% coverage)
5. Optimize slow tests to reduce execution time

Total Estimated Coverage After Week 5: 80-85%