IEEE Standard for Software and System Test Documentation

829-1998

TraceRecoviz

버전: 1.4 | 작성일: 2025-10-18 |

표준 근거: IEEE 829-2008, IEEE 1008-1987

1. Master Test Plan (MTP)

1.1 Introduction

1.1.1 Purpose

TraceRecoviz는 C++ 기반 단위 테스트(GoogleTest)의 실행 흐름을 자동으로 추적하여 함수 호출, 반환 및 Assertion 이벤트를 시퀀스 다이어그램으로 시각화하는 도구이다. 이 문서의 목적은 IEEE 829-1998, IEEE 1008-1987 문서 표준을 준수하여, TraceRecoviz 의 기능 및 비기능 요구사항 (SRS)에 대응하는 Test 계획, 범위, 환경 및 산출물 등을 정의하는 것이다.

1.1.2 Scope

이 Test 계획은 TraceRecoviz에 대한 모든 레벨의 Test 계획을 포함한다.

- 단위 테스트(Unit Test) : Clang LibTooling 계측 로직 및 GoogleTest 리스너 모듈의 기능 검증
- 통합 테스트(Integration Test) : 로그 생성 -> 파싱 -> 다이어그램 생성 프로세스에서 관찰되는 데이터 검증
- 시스템 테스트 (System Test) : 전체 시스템의 동작, 성능 검증

시험 결과를 통해 다음 목표를 달성한다.

- SRS에서 정의된 모든 기능 요구사항이 정확히 구현되었는지 확인
- 비기능 요구사항의 성능, 신뢰성, 호환성 준수 검증
- 결함 및 누락 항목 식별을 통한 시스템 품질 보증

본 계획은 TraceRecoviz 을 대상으로 적용한다.

1.1.3 References

IEEE Std 829-2008 – Standard for Software and System Test Documentation

IEEE Std 830-1998 – Software Requirements Specification

TraceRecoviz SRS v1.4 (2025-10-18)

1.1.4 Definitions, Acronyms, and Abbreviations

용어	정의
TraceRecoviz	**C++/GoogleTest 기반 함수 호출·반환·Assertion 추적 및 시퀀스 다이어그램 생성 도구**
Instrumentation (INST)	**Clang LibTooling을 사용하여 테스트 소스에 자동으로 추적 코드를 삽입하는 과정**
TraceListener	**GoogleTest 의 EmptyTestEventListener를 상속하여 테스트 생명주기 이벤트를 로그로 기록하는 모듈**
CALL / RETURN / ASSERTION_CALL	**로그 라인의 행동 유형 식별자 (함수 호출, 반환, Assertion 발생)**
GoJS GraphLinksModel	**시퀀스 다이어그램 JSON 모델 형식 (nodeDataArray, linkDataArray)**
FR	**Functional Requirement (기능 요구사항)**
NFR	**Non-Functional Requirement (비기능 요구사항)**
LTP / LTC / LTR/ LR	**Level Test Plan / Case / Report / Log – IEEE 829에서 정의하는 시험 문서 유형**

1.2 Details of the Master Test Plan

1.2.1 Test Items

구분	구성 요소	설명
INST (Instrumentation)	inject_trace_tool.cpp, Clang LibTooling 기반	C++ 함수의 진입/종료/반환/Assertion 호출을 자동 삽입하는 도구. 모든 함수 호출에 대해 trace_enter/trace_return 코드가 주입되는지 검증한다.
TraceListener	trace_listener.cpp	GoogleTest 이벤트 훅(OnTestStart, OnTestEnd 등)을 통해 로그를 생성하고 파일 입출력을 관리한다.
Logger / FileUtil	trace_logger.cpp, file_util.cpp	로그 파일 생성, 회전 정책, 파일 닫기 시점의 정합성을 시험한다.
Parser	trace_parser.py	로그를 정규식 기반으로 파싱하여 JSON 구조(GraphLinksModel)로 변환하는 기능을 시험한다.
Diagram Generator	diagram_gen.py	파서 출력(JSON)을 GoJS 다이어그램 모델로 시각화 가능한지 확인한다.

1.2.2 Features to be Tested

요구사항 ID	시험 항목	설명
FR-INST-001~005	함수 진입/종료 계측	모든 함수에 trace_enter, trace_return 이 정확히 삽입되는지, return 문이 없는 함수(생성자/소멸자 포함)에서 누락되지 않는지 시험한다.
FR-LOG-001~003	로그 파일 생성 및 포맷	테스트 케이스 단위로 로그 파일이 생성되고, [TRACE] 접두어와 표준 로그 포맷이 준수되는지 검증한다.
FR-FMT-001~003	로그 라인 정규식 일치성	파서 정규식(^₩[(?P<testname>.+?)₩] ...)과 완벽히 일치하는 로그가 생성되는지 확인한다.
FR-AST-001~002	Assertion 로깅	EXPECT_*, ASSERT_* 호출 시 대응되는 [ASSERTION_CALL] 로그가 기록되는지 시험한다.
FR-PAR-001~005	파서 기능	로그에서 caller/callee, 반환값, assertion 정보를 정확히 추출하고 GoJS JSON 구조를 생성하는지 시험한다.
FR-SDR-001~007	시퀀스 다이어그램 모델링	CALL-RETURN 대응하는지, 미종결 호출 처리, Assertion 시각화, 객체별 고유 ID 매핑을 검증한다.

1.2.3 Features Not to be Tested

- 환경(Windows MSVC, GCC 등)에서의 동작 보장은 본 계획 범위에서 제외한다.
- GoJS 구성 요소(렌더링 프레임워크 자체)의 내부 동작은 시험하지 않는다.
- GoogleTest 프레임워크 자체의 로깅 기능은 외부 컴포넌트로 간주하며 제외한다.

1.2.4 Test Approach

1. Unit Level (LTP-UNIT)
   - 단일 모듈(계측기, 리스너, 파서)에 대한 동작 검증
   - Clang Rewriter 후의 코드 구조, 로그 포맷 일치성, 함수 진입/반환 로그 일관성 테스트

- GoJS JSON 입력에 대한 시각화 정확성 검증
- Assertion 및 객체간의 메시지 노드 색상 매핑 테스트

2. Integration Level (LTP-INT)
   - 계측기 -> 리스너 -> 파서 간 데이터 흐름 검증
   - 실제 테스트 실행 후 생성된 로그를 파서에 투입하여 시퀀스 다이어그램 재구성 가능 여부를 확인
3. System Level (LTP-SYS)
   - 전체 도구의 동작과 성능 테스트

1.2.5 Item Pass / Fail Criteria

구분	기준
Pass	SRS의 정규식, 로그 형식, JSON 구조 정의에 모두 일치하며 오류나 누락이 없음
Fail	로그 포맷 불일치, 파서 오류, 시퀀스 불일치, assertion 이벤트 누락 등 발생 시

1.2.6 Suspension / Resumption Criteria

Suspension : 파서 정규식 불일치 또는 로그 파손으로 인해 시퀀스 다이어그램 생성이 불가할 경우

Resumption : 정규식 수정, 로그 포맷 보정 후 재시험 수행

1.2.7 Enviromental Needs

OS : Linux / Unix 계열

Compiler : Clang

Framework : GoogleTest 1.12 이상

Language : C++ 17

Tools : Python 3.9 이상, GoJS

1.2.8 Risks

- 로그 포맷 변경 시 기존 파서 정규식 불일치 위험
- GoogleTest 버전 업그레이드로 이벤트 훅 시그니처 변경 가능성
- 파서 처리량 저하 시 성능 시험 기준 미달 가능

2. Level Test Plan – Unit Testing (LTP-UNIT)

2.1 Introduction

이 장에서는 TraceRecoviz 시스템의 개별 구성 요소(Instrumentation, Listener, Logger, Parser)의 단위 기능 검증 계획을 정의한다.
단위 시험은 각 모듈의 독립적인 동작이 SRS에 명시된 기능 요구사항(FR)에 따라 정확히 수행되는지를 확인한다.
시험 환경은 개발 환경과 동일한 Linux + Clang LibTooling + GoogleTest 조합을 사용한다.

2.2 Details for this level of test plan

2.2.1 Test Items

모듈	소스 파일	시험 항목
Instrumentation	inject_trace_tool.cpp	함수 진입·반환 계측 삽입 (FR-INST-001 ~ 005)
TraceListener	trace_listener.cpp	테스트 시작/종료 로그 및 Assertion 이벤트 기록 (FR-LOG-001 ~ 003, FR-AST-001 ~ 002)
Logger	trace_logger.cpp	로그 파일 생성, 열기/닫기 동작 검증 (FR-LOG-001)
Parser	trace_parser.py	로그 정규식 일치성 검증 (FR-FMT-001 ~ 003)

2.2.2 Features to be Tested

요구사항 ID	시험 내용	기대 결과
FR-INST-001	함수 시작부에 trace_enter() 삽입	모든 함수 진입점에서 호출 로그 생성
FR-INST-002	모든 return 문에 trace_return() 삽입	반환 로그 존재 및 값 일관성 유지

요구사항 ID	시험 내용	기대 결과
FR-INST-003	명시적 반환 없는 함수(생성자/소멸자) 처리	블록 말미 자동 trace_return() 호출
FR-LOG-001	테스트 실행 시 로그 파일 생성 확인	build/log/*.log 경로에 생성됨
FR-LOG-002	생명주기 이벤트 [TRACE] 라인 기록	시작/종료 메시지가 포함됨
FR-AST-001	Assertion 발생 시 [ASSERTION_CALL] 로그	EXPECT/ASSERT 호출 모두 로깅
FR-FMT-001	로그 라인 정규식 일치 여부	모든 로그가 파서 정규식과 매칭됨

2.2.3 Approach

단위 시험은 GoogleTest 기반 자동화 테스트로 수행한다.

1. Instrumentation Test
- 입력: 간단한 함수가 포함된 C++ 코드
- 절차: inject_trace_tool 실행 후 출력된 코드 분석
- 검증: trace_enter, trace_return 삽입 여부 비교
2. Listener + Logger Test
- 입력: GoogleTest 단위 테스트 코드
- 절차: 테스트 실행 후 로그 파일 생성 여부 확인
- 검증: [TRACE], [CALL], [RETURN], [ASSERTION_CALL] 라인 존재 여부
3. Parser Test (정규식 단위 검증)
- 입력: 표준 로그 샘플
- 절차: trace_parser.py 실행
- 검증: 각 라인이 정규식 그룹(testname, action, caller_sig 등)에 정확히 매핑되는지

2.2.4 Entry / Exit Criteria

구분	기준
Entry	각 모듈이 컴파일 + 빌드 완료된 상태
Exit	모든 단위 테스트 케이스가 Pass, 주요 오류 0건 이하

2.2.5 Pass / Fall Criteria

상태	기준
Pass	로그 정규식 100% 일치 및 함수 계측 오류 없음
Fail	로그 포맷 불일치, 파일 미생성, Assertion 누락 발생

2.2.6 Enviromental Needs

OS : Ubuntu 20.04 이상

Compiler : Clang

Framework : GoogleTest + Python

의존 라이브러리 : libclang, abi::__cxa_demangle

2.2.7 Risks

템플릿 함수/오버로드 함수의 시그니처 치환실패

테스트 환경(경로 권한) 문제로 로그 파일 미생성

3. Level Test Plan – Integration Testing (LTP-INT)

3.1 Introdiction

통합 테스트는 TraceRecoviz의 핵심 흐름인 계측 -> 리스너 -> 로거 -> 파서 -> 시퀀스 다이어그램 생성까지의 단계가 정상적으로 연동되는 지를 검증한다. 각 모듈 간 데이터 형식과 로그의 변환이 주 평가 대상이다.

3.2 Details for this level of test plan

3.2.1 Test Items/Interfaces

단계	관련 모듈	검증 항목
1	Instrumentation → Listener	계측된 코드가 리스너에 정상적으로 로그를 전달하는가
2	Listener → Logger	테스트 단위별 로그 파일이 자동 생성·종료되는가
3	Logger → Parser	로그 파일이 정규식에 완벽히 부합하는가
4	Parser → Diagram Generator	파서 출력(JSON)이 GoJS 모델 형식을 충족하는가

3.2.2 Features to be Tested

요구사항 ID	시험 내용	기대 결과
FR-LOG-001 ~ 003	테스트 수트 실행 → 로그 파일 연결성 검증	각 테스트 시작/종료 시점에 정상 로그 기록
FR-PAR-001 ~ 005	로그 입력 → JSON 출력	caller/callee, return 값이 올바르게 매핑
FR-SDR-001 ~ 007	전체 시퀀스 재구성 검증	CALL-RETURN 대응 100%, Assertion 시각화 정확

3.2.3 Approach

1. **End-to-End 실행 테스트**

- 입력 : 계측된 GoogleTest 프로젝트
- 절차 : 테스트 전체 실행 -> .log 생성 -> 파서 -> JSON 출력
- 검증 : 로그 -> JSON -> 다이어그램 간 이벤트 수 및 순서 일치

2. **Log Test**

- 이전 버전 로그와 신규 로그 비교
- Diff 도구로 변경 마커 (+, -, =, !) 색상 매핑 검증

3.2.4 Entry/Exit/Pass

항목	기준
Entry	모든 단위 시험 성공 및 빌드 성공
Exit	통합 플로우에서 데이터 손실 또는 오류 0건

4. Level Test Plan – System Testing (LTP-SYS)

3.1 Introduction

시스템 테스트는 TraceRecoviz의 전체 흐름(Instrumentation -> Listener -> Logger -> Parser -> Diagram Generator)이 실제 운영 환경에서 하나의 통합 시스템으로 정상 동작하는지 검증한다.

4.2 Details for this level of test plan

4.2.1 Test Items

구분	설명
전체 도구 흐름	소스 계측부터 시퀀스 다이어그램 렌더까지 End-to-End 검증
성능 측정 모듈	로그 라인 수 대비 파서 처리 속도 평가
복원 테스트 로직	RETURN 누락, 비정상 종료 로그 등의 복구 정확성 검증

4.2.2 Features to be Tested

요구사항 ID	시험 내용	기대 결과
NFR-PERF-001	계측 삽입 시간 측정	기준 대비 컴파일 시간 30 % 이내
NFR-PERF-002	테스트 수행 시 추적 오버헤드 측정	평균 5 ms 이내
NFR-PERF-003	파서 처리 성능 검증	5만 라인/초 이상 처리
NFR-REL-001	로그 정합성 검증	CALL/RETURN 쌍 누락 0건
NFR-REL-002	trace_return 정확성	모든 함수에서 1회 호출 보장
FR-SDR-008	부분 로그 복원 검증	RETURN 누락 시 미종결 마커 출력
FR-SDR-009	순서 복원 검증	단조 증가 시퀀스 번호로 정렬 시 순서 일치

4.2.3 Approach

1. Stress & Load Test

- 입력 : 대규모 테스트 프로젝트 (함수 300개 이상)

- 측정 : 로그 생성 시간, 파일 크기, 파서 처리 속도

- 기준 ; CPU 70% 이하, 메모리 2GB 이내 사용

2. Fault Injection Test

- 로그에 임의 라인 손실/오류 삽입

- 검증 : 파서가 다이어그램을 생성할 때 누락 라인 표시

3. Reliability Test

- 100회 연속 실행 테스트

- 기존 : Crash 0회, 로그 손실 없음

4.2.4 Entry / Exit Criteria

항목	기준
Entry	모든 통합 시험 성공 및 파서/렌더러 빌드 완료
Exit	성능·신뢰성 지표 모두 목표 기준 달성, 결함 심각도 High 이하

4.2.5 Pass / Fail Criteria

상태	판정 기준
Pass	SRS의 성능 · 신뢰성 · 복원성 요구 모두 충족
Fail	오버헤드 초과, 데이터 손실, 로그 불일치 발생

4.2.6 Enviromental Needs

- **OS : Ubuntu 20.04**
- **CPU : 5-core 이상, RAM 16GB 이상**
- **Clang, GoogleTest, Python 3.9 + GoJS 환경**
- **테스트 디렉토리 : build/log/, output/json/, output/img**