Software Requirements Specification

도구명 : TraceRecoviz

C++ 실행 추적 및 시퀀스 다이어그램 생성 도구

버전 : 1.4

작성일 : 2025-10-18

목차

1 Introduction

1.1 Purpose

본 문서는 C++/GoogleTest 테스트 실행 시 함수 호출/반환 및 Assertion을 로그로 수집하고, 이를 파싱하여 시퀀스 다이어그램(GoJS 버전)으로 변환하는 도구 모음(계측코드 삽입 도구, GTest 리스너, 로거/파서, 파일 유틸)의 요구사항을 정의한다. GTEST 기반의 C++ 테스트 코드를 실행했을 때 TEST CASE 단위로 발생한 모든 호출을 하나의 시퀀스다이어그램으로 표시하는 것이 목표이며, 이를 위해 명확한 객체, 호출 동안에 발생하는 런타임상의 정확한 값과 타입, 호출의 대상과 호출명을 기록을 해야한다. IEEE Std 830-1998이 권장하는 구조를 따른다.

1.2 Scope

- 계측 도구 : Clang LibTooling 기반으로 C++ 함수의 진입/종료, return 문, assertion 매크로를 자동 삽입하고 치환한다. 최종적으로 변환된 코드는 표준 출력을 내보내는 역할을 한다.
- 실행 추적기/리스너 : GoogleTest 이벤트 훅을 통해서 테스트 생명주기 로그를 남기고, 테스트 시작시 파일을 열며 종료 시 닫는다.
- 로그 형식 및 저장 : build/{TRACE_VARIANT}/<파일명>.log 경로에 기록 (Default : TRACE_VARIANT="log")
- 파서/다이어그램 생성기 : 정규식 패턴으로 로그를 해석하여 GoJS GraphLinksModel JSON 을 생성한다.

1.3 Definitions, Acronyms, and Abbreviations

- 계측 삽입 : 소스에 추적 코드를 자동 삽입하는 과정
- TraceListener : GTest 이벤트 리스너 클래스
- CALL/RETURN/ASSERTION_CALL : 로그 라인 액션 구분자(정규식 그룹)
- GoJS GraphLinksModel : GoJS 시퀀스 다이어그램 데이터 모델 키 (nodeDataArray, LinkDataArray)

1.4 References

- IEEE Std 830-1998 : 섹션 구성, 외부 인터페이스, 기능 요구사항, 품질 속성 등 템플릿-가이드라인

1.5 Overview

Section 2에서는 도구 개요, Section 3에서는 구체적인 요구사항(외부 인터페이스, 기능, 성능, 제약 속성)을 제공한다.

2 Overall description

2.1 Product perspective

본 소프트웨어는 소스 계측, 실행 시의 리스너/로거, 로그 파서로 구성된 모듈 묶음이다. 소스 계측은 Clang ASTFrontendAction/ASTConsumer 기반으로 동작하며, 수정된 코드를 표준 출력으로 제공한다. 실행 시에는 GTestEmptyTestEventListener를 통해 테스트 생명주기를 캡처한다.

2.2 Product functions

- 함수 진입(trace_enter)/반환(trace_return) 로깅 및 모든 return 경로 래핑 삽입, void/ctor/dtor 반환 처리
- EXPECT_* / ASSERT_* 호출을 _LOG 형태로 치환하거나 전처리를 통해 로그를 남김.
- 테스트 파일명 / 테스트 스위트 / 테스트 케이스 기반으로 로그 파일 열기 / 닫기
- 표준화된 로그 라인 형식 (CALL/RETURN/ASSERTION_CALL)과 파서 정규식을 구성
- 시퀀스 다이어그램 JSON 생성(GoJS 모델)

2.3 User characteristics

- C++ / CMake / GTest 을 사용하는 개발자, CI 엔지니어, 테스트 자동화 담당자

2.4 Constraints

- 컴파일러 / 툴체인 : Clang LibTooling 필요 (코드 내 Clang 전방동작 / ReWriter 사용)
- 표준 : 빌드 플래그 예시로 -std=c++17이 사용
- 테스트 프레임워크 : GoogleTest 이벤트 리스너를 전제

2.5 Assumptions and dependencies

- 로그 파일은 build/{TRACE_VARIANT}/ 하위에 생성(.log 형식)

- 파서가 기대하는 로그 포맷은 정규식에 정의된 형식과 일치해야 한다.

3 Specific requirements

3.1 External Interfaces

3.1.1 User Interfaces

- **상단 바 (①)**

  1.1. 시스템은 상단 바에 응용 프로그램 명칭 **"TraceRecoviz"** 을 표시하여야 한다.

  1.2. 상단 바는 종료 버튼을 포함하여야 한다.

-

- **좌측 제어 패널 (②)**
- 2.1. 제어 패널은 "프로젝트", "동작", "편집"의 세 섹션으로 구성되어야 한다.
- [프로젝트 세션]

  2.2. 시스템은 "수정 전 프로젝트 경로" 및 "수정 후 프로젝트 경로"의 두 개 입력 필드를 제공하여야 한다.

2.3. 각 입력 필드 우측에는 "폴더 선택" 버튼이 배치되어야 하며, 버튼 클릭 시 폴더 선택 대화상자를 통해 사용자가 디렉터리를 지정할 수 있어야 한다.

2.3. 경로 필드의 내용은 사용자가 직접 수정할 수 없다.

- [동작 세션]

2.4. 제어 패널에는 "빌드 및 실행" 버튼과 "웹서버" 버튼이 나란히 배치되어야 한다.

2.5. 빌드 및 실행" 버튼을 클릭하면 시퀀스 다이어그램을 생성하는 일련의 동작을 수행한다.

2.6. 빌드 과정 중 표준 출력 및 오류 메시지는 별도의 로그 팝업 창에 실시간으로 표시되어야 한다.

2.7. "웹서버" 버튼은 백그라운드 웹서버의 동작 상태를 제어하는 ON/OFF 토글 버튼으로 작동하여야 한다.

- [편집 섹션]

2.8. "Makefile 편집" 버튼이 포함되어 있으며, 클릭 시 Makefile 편집 창이 실행된다.

2.9. 편집 창에서는 프로젝트 디렉터리 내의 Makefile을 읽어와 내용을 수정·저장할 수 있다.

- **우측 메인 작업 영역 (③)**
- 3.1. 우측 영역은 폴더 트리, 파일 목록, 미리보기, 로그 콘솔로 구성되어야 한다.
- [폴더 트리 영역]

3.2. 폴더 트리는 선택한 프로젝트 디렉터리를 기준으로 폴더 구조를 계층적으로 표시한다.

3.3. 사용자가 특정 폴더를 선택하면, 그 하위 파일 목록이 자동으로 파일 목록 창에 표시되어야 한다.

- [파일 목록 영역]
- 3.4. 파일 목록 창은 선택된 폴더의 모든 파일을 목록 형식으로 표시하여야 하며, 컬럼에는 파일명, 크기, 수정일, 형식 등 정보가 포함되어야 한다.

3.5. 사용자가 파일을 클릭하면 해당 파일의 상세 내용이 미리보기 창에 표시되어야 한다.

- [미리보기 영역]
- 3.7. 미리보기 창은 선택된 선택된 파일의 유형을 자동 판별하여 적절한 표시 방식을 적용한다.
  3.8. 텍스트 파일은 수정 불가능한 텍스트로 표시되어야 한다.
  3.9. 기타 파일은 파일명, 경로, 크기 등의 기본 정보만 표시되어야 한다.
- [로그 콘솔 영역]
- 3.10. 로그 콘솔은 프로그램 동작 중 발생한 이벤트에 대해 실시간으로 표시하여야 한다.

3.1.2 Hardware Interfaces

- OS가 리눅스 또는 windows 환경이어야 한다.
- GUI(Graphical User Interface) 환경을 지원해야 하며, 사용자는 마우스 및 키보드를 통해 시스템과 상호작용할 수 있어야 한다.

3.1.3 Software Interfaces

- **Clang LibTooling**
- Name : LLVM Clang LibTooling Framework
- Version Number : 18.1.6 (or compatible with LLVM 18 release)
- Source : https://clang.llvm.org/docs/LibTooling.html
- Description : C/C++ 코드의 함수 호출 계측을 위해 Clang LibTooling을 사용한다.
- **PySide6 (Qt for Python)**
- Name : PySide6 GUI Framework
- Version Number : 6.7.1
- Source : https://doc.qt.io/qtforpython/
- Description : 사용자 인터페이스(UI)는 PySide6를 기반으로 구현한다.
- **FastAPI**
- Name : FastAPI Web Framework
- Version Number : 0.110.2

- Source : https://fastapi.tiangolo.com/
- Description : 내장 웹서버 기능으로 생성한 시퀀스다이어그램을 전시에 사용한다.
- **GoogleTest**
- Name : GoogleTest C++ Unit Testing Framework
- Version Number : 1.15.0
- Source : https://github.com/google/googletest
- Description : 테스트의 생명주기를 계측하기 위해 활용한다.
- **Make Build System**
- Name : GNU Make Build System
- Version Number : 4.3
- Source : https://www.gnu.org/software/make/manual/
- Description : GNU Make를 사용하여 프로젝트를 빌드한다.
-

3.1.4 Communications Interfaces

- **네트워크 통신 없음.**

3.2 Functional Requirements

3.2.1 Clang 기반의 계측(Instrumation, INST)

- **FR-INST-001**
  함수 시작부에 trace_enter((this 또는 0), __PRETY_FUNCTION__, 파라미터 등)을 삽입해야 한다.
- **FR-INST-002**
  모든 return 문을 래핑하여 trace_return 호출 후 원래 반환값을 반환해야 한다. 참고 반환도 포함된다.
- **FR-INST-003**
  Void/생성자/소멸자 에서 명시적인 return 이 없는 경우, 블록 말미에 trace_return을 삽입해야 한다.
- **FR-INST-004**
  EXPECT_*, ASSERT_* 매크로는 대응하는 *_LOG 형태로 치환되어야 한다.

- **FR-INST-005**
  매크로 토큰 감지시 직접 [ASSERTION_CALL] ... 로그를 삽입하고 출력할 수 있어야 한다.

3.2.2 로깅 형식

- **FR-LOG-001**
  테스트 시작 시 <소스파일명>_<스위트>_<테스트>.log 이름으로 로그를 열고, 종료 시 닫아야 한다.
- **FR-LOG-002**
  생명주기 이벤트(프로그램/이터레이션/스위트/테스트/테스트 환경 세팅 및 해제 를 [TRACE] 라는 접두어로 기록해야 한다.
- **FR-LOG-003**
  로그 파일 경로는 build/{TRACE_VARIANT}}/ 하위에 생성되어야 하며, 기본값은 TRACE_VARIANT="log" 이다.

3.2.3 로그 라인 포맷-시그니처 정규화

- **FR-FMT-001**
  trace_enter는 **[CALL] |caller=<ptr>| <caller_sig> >> |callee=<ptr>| <callee_sig> [|ARGS|(...)]** 형식으로 한 줄을 남겨야 한다.
- **FR-FMT-002**
  trace_return은 **[RETURN] |caller=<ptr>| <caller_sig> >> |callee=<ptr>| <callee_sig> => <ret>** 형식을 따라야 한다. (return 부분의 포맷은 trace_return 구현 상단/콜스택 처리에 의해 결정)
- **FR-FMT-003**
  템플릿 함수의 시그니처 내 템플릿 매개변수명은 실제 매핑으로 치환하고 [...] 표기를 제거해야 한다.
- **FR-FMT-004**
  타입 이름은 abi::__cxa_demangle 을 통해 사람이 읽기 쉬운 문자열 기록해야 한다.

3.2.4 Assertion 로깅

- **FR-AST-001**
  LOG_ASSERTION 매크로는 현재 테스트명과 함께 [ASSERTION_CALL] 라인을 출력해야 한다.

- **FR-AST-002**
  EXPECT_*_LOG, ASSERT_*_LOG 매크로는 원래 Assertion 문을 그대로 실행하면서, 동일한 라인을 사전에 로깅해야 한다.

3.2.5 파싱 및 다이어그램 생성

- **FR-PAR-001**
  파서는 다음 정규식과 완전히 호환되는 로그만 수용한다. 정규식에는 다음과 같은 내용이 담겨 있어야 한다.
  1. 실행되는 test의 이름
  2. 현재 호출의 타입(추가, 제거, RETURNE, ASSERT) 중에서 하나
  3. Caller의 주소와 Caller의 타입
  4. Callee의 주소와 Callee의 타입
  5. 인자값과 반환값의 값과 타입
  6. 정규식 형식
     ^₩[(?P<testname>.+?)₩]₩s+₩[(?P<action>CALL|RETURN|ASSERTION_CALL)₩]₩s*₩|caller=(?P<caller_ptr>[^|]+)₩|₩s*(?P<caller_sig>.+?)₩s*>>₩s*(?:₩|callee=(?P<callee_ptr>[^|]+)₩|₩s*(?P<callee_sig>.+?))?(?:₩s*=>₩s*(?P<return_val>.+))?$
- **FR-PAR-002**
  파서는 캡처한 caller/callee의 반환형, 클래스, 함수명을 parse_caller_sig/parse_callee_skg로 나눠야 한다.
- **FR-PAR-003**
  결과는 GoJS GraphLinksModel JSON 사양 (nodeDataArray, linkDataArray)을 만족해야 한다.
- **FR-PAR-004**
  라인 선두의 변경 마커(+, -, =, !)는 링크 색상 메타데이터(green, red, yellow)로 매핑되어야 한다.
- **FR-PAR-005**
  노드 최초 등장 시 key/text/loc/size/isGroup/duration 등의 필드를 초기화해야 한다.

3.2.6 파일 유틸

- **FR-FILE-001**

JSON 결과 저장 시 폴더가 없으면 생성 후 저장 로그를 남겨야 한다.

- **FR-FILE-002**
  특정 확장자에 대해 일괄 삭제 기능을 제공해야 한다.

3.2.7 추적 로그

- **FR-SDR-001**
  시퀀스다이어그램을 그릴 때 각 호출 이벤트마다 모든 매개변수에 대해 런타임 타입 식별자와 값 표현을 기록해야 한다. 또한 스칼라/포인터/객체 인자가 섞인 호출 1건에 대해, 매개변수 수와 동일한 개수의 (타입, 값/ID) 항목이 로그에 존재한다.
- **FR-SDR-002**
  시퀀스다이어그램을 그릴 때 각 완료된 호출에 대해 반환 타입 식별자와 반환 값 표현을 기록해야 한다. 생성자/소멸자의 경우에는 반환 값 대신 <constructed>,<destroyed>와 같이 구별 가능한 토큰을 기록해야 한다. 반환이 있는 함수는 (타입, 값/ID) 가 기록되고, 생성자/소멸자는 지정 토큰으로 기록된다.
- **FR-SDR-003**
  시퀀스다이어그램을 그릴 때 실행 동안 각 객체 인스턴스에 대해 고유하고 안정적인 런타임 ID를 부여·기록해야 한다. 동일 클래스의 서로 다른 인스턴스는 반드시 다른 ID를 가져야 한다. 클래스 인스턴스 A, B가 서로를 호출하는 시나리오에서, 다이어그램에 서로 다른 두 개의 라이프라인이 생성된다.
- **FR-SDR-004**
  시퀀스다이어그램을 그릴 때 프로그램 가시 범위의 모든 호출에 대해 정확히 1개의 CALL과 이에 대응하는 1개의 RETURN을 기록해야 하며, 발생 순서를 재구성할 수 있어야 한다(예: 단조 증가 시퀀스 번호 또는 타임스탬프). 중첩 호출 m건에 대해 로그에는 m개의 CALL과 m개의 RETURN이 존재하고, 정렬 시 호출 전·후 관계가 보존된다.
- **FR-SDR-005**
  시퀀스다이어그램을 그릴 때 테스트 실행의 단계 경계를 로그에 명시해야 한다. 최소한 (a) 전역/수트/테스트 시작·종료, (b) Fixture SetUp/TearDown, (c) 테스트 본문(Test Body) 의 경계가 식별 가능해

야 하며, 파라미터화된 테스트의 경우 파라미터 표식을 포함해야 한다. 테스트 케이스에 대해 "전역/수트/테스트 시작 → SetUp → 본문 호출들 → TearDown → 종료" 경계 표식이 모두 존재하고, 다이어그램에서 단계별 구분(예: 레이블/스윔레인/주석 중 하나)로 식별 가능하다.

- **FR-SDR-006**
  시퀀스다이어그램을 그릴 때 각 assertion에 대해 assertion 이벤트를 기록해야 하며, 표현식 요약과 결과(통과/실패) 를 포함해야 한다. assertion이 3회 발생하는 테스트에서, 로그에는 3개의 assertion 이벤트가 존재하고 결과 값이 식별된다.
- **FR-SDR-007**
  시퀀스다이어그램을 그릴 때 유효한 로그로부터 시퀀스 다이어그램 모델을 생성할 수 있어야 하며, 다음이 충족되어야 한다:
  (a) 노드 ↔ 런타임 ID가 1:1로 대응,
  (b) 에지 ↔ CALL/RETURN가 1:1로 대응,
  (c) 생성자/소멸자는 라이프라인 시작/종결 이벤트로 시각화 가능,
  (d) 테스트 단계 경계가 다이어그램 상에서 식별 가능.
  표준 샘플 로그를 투입하면 생성된 모델에서 (a)~(d)가 모두 충족된다.
- **FR-SDR-008**
  시퀀스다이어그램을 그릴 때 RETURN이 누락된 호출(예: 크래시/타임아웃)이 존재해도 부분 다이어그램을 산출해야 하며, 해당 호출을 미종결 상태로 표식해야 한다. 의도적으로 RETURN을 생략한 시나리오에서, 산출물에 "미종결 호출" 마커가 포함된다.
- **FR-SDR-009**
  시퀀스다이어그램을 그릴 때 이벤트 간 **전역 순서**를 재구성할 수 있는 정보를 **각 레코드에 포함**해야 한다(예: 단조 증가 시퀀스 번호). 로그를 해당 키로 정렬했을 때 호출-반환 순서가 위배되지 않는다.

3.3 Performance Requirements

- **NFR-PERF-001**
  계측 삽입으로 인한 컴파일 시간 증가는 기준 대피 30%를 넘지 않아야 한다.

- **NFR-PERF-002**
  실행 시 추적 오버헤드는 테스트 케이스당 평균 5ms 이내여야 한다. (릴리즈 빌드 기준)
- **NFR-PERF-003**
  파서의 단일 로그 파일 처리 시간은 1만 라인/초 이상이어야 한다.

3.4 Design Constraints

- **NFR-CON-001**
  Clang LibTooling 및 C++ 17 사용, 컴파일 플래그 -std=c++17
- **NFR-CON-002**
  GoogleTest 의존
- **NFR-CON-003**
  로그 경로 구조는 build/{TRACE_VARIANT}/로 고정되어야 하며 환경 변수 / 매크로로 가변화 가능

3.5 Software System Attributes

3.5.1 Reliability

- **NFR-REL-001**
  로그 라인의 액션/시그니처/포인트 필드는 누락없이 출력되어야 한다. 정규식으로 검증 가능하다.
- **NFR-REL-002**
  모든 경로에서 trace_return이 정확히 1회 호출되어야 한다.

3.5.2 Availablility

- **NFR-AVL-001**
  계측 삽입은 소스 수정 후 표준 출력으로 기능해야 한다.
- **NFR-AVL-002**
  Assertion 로깅은 매크로 치환 테이블로 관리되어야 한다. 항목이 추가가 될 수 있다.

3.5.3 Porability

- **NFR-PROT-001**
  리눅스/유닉스 계열 Clang 환경에서 동작을 보장한다. Clang/abi demangel을 의존한다.

3.5.4 Maintainability

- **NFR-MNT-001**
  콜스택 관리는 스레드 로컬 스택으로 수행되어야 한다.
- **NFR-MNT-002**
  생성자/소멸자 판별 로직을 통해 해당 로그의 의미를 올바르게 부여해야 한다.

4 부록

4.1 데이터 사전

- testname : 테스트 식별자, 테스트스위트.테스트명 형식
- action : CALL | RETURN | ASSERTION_CALL
- caller_ptr / caller_sig : 호출자 주소/시그니처
- callee_ptr / callee_sig : 피호출자 주소/시그니처
- return_val : 반환값 텍스트, 정규식 정의 참조

4.2 파일 구조 및 경로 규칙

- 로그 저장 경로 : build/{TRACE_VARIANT}/<파일명>.log
- 테스트 시작 시 파일 명 규칙 : <소스파일>_<스위트>.<테스트>.log

4.3 사용 시나리오

1. 계측 도구 실행 -> 변환 소스 생성(표준 출력> -> 빌드
2. 테스트 실행(리스너 자동 등록) -> 로그파일 생성, 기록, 종료
3. 파서로 고르 읽기 -> GraphLinksModel JSON 산출 -> 시퀀스 다이어그램 렌더

4.4 IEEE 830 템플릿

본 SRS는 IEEE 830 권장 섹션 및 외부 인터페이스, 성능, 제약, 시스템 속성 항목 구성을 따랐다.