열차제어시스템 소프트웨어 안전성 평가기법
Safety Assessment of Train Control System Software

한재중* 조병태** 황종규*** 조현정*** 김형신**
Jaejoong Han Byungtae Cho Jonggyu Hwang Hyunjeong Jo Hyungshin Kim

ABSTRACT

Recently, train control system is adopting computer system replacing mechanical system and its software is taking more responsibility than ever. Train control system software is a safety-critical embedded software with realtime and high reliability requirements. In this paper, we propose a safety assessment method for the train control system software. We review characteristics of train control system software and analyze related international software safety standards to derive requirements for safety assessment. Testing tools used for embedded software are surveyed to find a feasible safety assessment architecture. The proposed safety assessment method is to use safety activity results generated during development processes and feed them to the runtime embedded software testing tool.

1. 서 론

열차제어 시스템은 최근 기존의 기계적 장치로부터 컴퓨터시스템으로 전환되고 있으며, 소프트웨어에의 의존성이 급격하게 증가하고 있다. 대표적인 시스템으로는 일본의 EJTC[1] 자동열차제어시스템(ATC)을 들 수 있다. EJTC의 ATC는 Level 0의 선로변 신호교환을 통한 차량제어로부터 Level 3의 무인 전자동 차량제어시스템으로 구성되어 있다. 특히 Level 3 ATC 시스템은 기존의 트랜스폰더 방식이 아닌 일반 무선통신방식을 이용하여 차량과 역관제소와의 통신에 기반하는 통신기반 열차제어 시스템(CBTC, Commnunication-based Train Control)이다. 이 시스템은 열차의 위치정보를 외부기기의 도움없이 차상기기들만을 이용하여 자체적으로 결정하고 있으며, 이를 이용하여 자동열차보호(ATP)기능을 수행할 수 있도록 설계되어 있다. 스웨덴에서도 ATP 기능을 수행하는 3중 중복 컴퓨터(TMR)을 ATC를 위해 차상에 장착하여 사용해오고 있다[2].

이와같이 초기의 기계적, 수동 차량신호시스템에서부터 최근의 무인 자동 열차제어 시스템으로 변천되어 오면서, 다수의 컴퓨터들이 차상장치로 사용되기 시작했으며, 이들 컴퓨터에 탑재되는 소프트웨어의 신뢰성과 안전성을 검증하는 것이 중요한 문제도 대두되기 시작했다. 소프트웨어의 안전성은 주로 소프트웨어의 개발초기 단계인 소프트웨어 설계 단계에서 안전성 활동을 수행함으로써 이루어진다. 대표적인 안전성 활동으로는 HAZOP[5], FTA[3, 4], FMECA[6] 등을 들 수 있다. 소프트웨어의 개발 초기에는 이와 같은 기법들이 활동되고 있으나, 개발이 완료된 이후에는 추가적인 안전성에 대한 검증은 자동화되거나 정형화된 방법이 적용되지 않고 있다. 소프트웨어의 안전성을 인증하는 경우에는 피인증기관의 안전성 활동을 평가할 때, 피인증기관이 제공하는 문서에 의존하여 평가하는 경우, 소프트웨어 안전성 활동의 충실성을 파악하기가 어려우며, 경우에 따라서는 추가적인 검증이 필요하게 된다. 이런

* 충남대학교, 임베디드 시스템 연구실
E-mail : pdjj14@cnu.ac.kr
TEL : (042) 821-7446 FAX : (042) 822-9959
** 충남대학교
*** 한국 철도 기술 연구원

경우 자동으로 소프트웨어의 안전성을 평가할 수 있는 도구가 있다면, 평가의 신뢰성을 높일 수 있어 큰 도움이 된다.

본 연구에서는 열차제어 시스템 소프트웨어의 안전성을 평가하는 방법을 제안한다. 열차제어시스템 소프트웨어의 특성을 분석하고, 안전성 관련 국제표준들을 조사하여 열차제어시스템 소프트웨어의 안전성 평가를 위한 요구조건을 도출하였다. 도출한 요구조건을 적용하기 위하여 기존의 임베디드 소프트웨어의 시험을 위해 제안된 도구들을 검토하였으며, 소프트웨어의 안전성을 평가할 수 있는 새로운 도구의 구조를 제안한다. 제안된 안전성 평가방법은 소프트웨어 설계과정에서 수행한 안전성활동의 결과를 이용하여 안전성관련 시험항목을 자동으로 생성하고, 그 결과를 동적 소프트웨어 시험 모듈에 적용하도록 하는 방법이다.

2. 열차제어 시스템 소프트웨어

열차제어 시스템은 초기의 기계장치를 이용하거나 수신호에 의존하던 원시적인 형태를 벗어나 무인 전자동화된 열차제어시스템(ATC)으로 진화하고 있다. 본 절에서는 통신기반 열차제어시스템(CBTC, Communication-based Train Control)과 같은 무인자동열차의 차상 탑재 컴퓨터에서 이용되는 소프트웨어의 특성을 분석하였다.

IT 기술이 발달함에 따라 차상시스템도 다수의 컴퓨터를 장착하게 되었으며, 이들 탑재컴퓨터들은 분산 네트워크를 구성하며 차상업무를 분담하여 처리하고 있다. 차상 장치에는 열차제어 컴퓨터(CBTC Trainborne Computer), 자동열차운영(ATO) 컴퓨터, 트랜스폰더 안테나, RF 통신장치, 차상랜, 속도센서 등이 있으며, 이들은 역으로부터 전송받은 명령을 해독하고, 그 결과에 따라 열차를 제어한다. 차상 컴퓨터들은 대부분 이중 이상의 중복모듈(dual redundancy)을 장착하고 있으며, 네트워크들도 유무선 이중중복 네트워크를 차상장치들에 제공하여 통신 신뢰성을 확보하고 있다.

CBTC 기반의 열차제어시스템의 주요 시스템 기능은 다음과 같다. 차상컴퓨터는 지상 관제국으로부터의 명령을 수신하여 검증하며, ATO 차상 컴퓨터와 연결되어 검증된 명령에 의한 열차제어 정보를 송신한다. ATO 차상컴퓨터는 열차의 추력기와 제동장치를 CBTC 차상컴퓨터로부터 수신한 속도, 가속도 값에 맞추어 제어하고, 기차역 플랫폼에서의 정차 제어기능을 제공한다. 이를 위하여 트랜스폰더를 통하여 정확한 열차위치 정보를 수신한다. CBCT 시스템과 같은 무인자동열차시스템에서는 ATO 기능을 주 운영모드로 활용하게 되며, 지상 ATO 장치들과 교신을 통해 열차제어를 하게 된다. 자동열차보호(ATP) 기능은 ATO 기능에 오류가 발생한 경우, 기관사의 열차제어를 보조하는 형태로 사용된다. 이외에도 문개폐 기능, 승객정보서비스, 자동시스템감독(ATS) 기능등이 있다.

차상 탑재 컴퓨터의 소프트웨어는 열차운행의 자동화와 자율화 추세에 따라 그 역할이 더욱 중요해지고 있으며, 따라서 소프트웨어가 열차제어 시스템 전체에 미치는 영향도 커지고 있다. 차상 탑재 컴퓨터는 마이크로 프로세서 시장의 급속한 발달에 따라 점차 고성능화 하고 있으며, 사용하는 프로그래밍 언어도 단순한 어셈블리어로부터 최근 상위수준 언어인 Ada 가 이용되기도 하였다[2]. ATC 소프트웨어는 스웨덴 Teknogram 사의 경우 1만 라인의 어셈블리어, 약 30 킬로바이트의 바이너리 이미지를 갖는다[2]. 열차제어 시스템소프트웨어의 크기와 복잡도는 하드웨어의 발달 속도보다는 느리지만, 점차적으로 규모가 커지며, 복잡도도 증가할 것을 예상된다.

이러한 열차제어 시스템 소프트웨어는 일종의 분산 임베디드 소프트웨어로 다음의 특성을 요구한다.

1) 실시간성 : 차상 컴퓨터는 지상 관제국으로부터 전송되는 명령을 정해진 시간내에 검증하고, 차내 네트워크를 통하여 제어 명령을 보내야 한다. 제어명령은 ATO 차상 컴퓨터의 출력 단자를 통하여 짧은 반응시간내에 제동장치 또는 동력장치에 전달되어야 대형 사고를 피할 수 있게 된다. 따라서 소프트웨어는 정해진 시간내에 기능들을 수행해야 하며, 또한 실행속도는 예측성(predictability)을 보장해야 한다.

2) 안전성 : 열차제어 시스템 전체의 안전성을 보장하기 위하여 소프트웨어의 안전성도 확보되어야 한다. 소프트웨어를 이용한 하드웨어 안전 기능장치(safety function)의 제어와 소프트웨어 자체의 개발과

정에서의 안전성 활동을 통한 시스템 안전성확보가 중요 요건이다.
3) 신뢰성 : 열차제어 시스템 전체의 신뢰성 및 안전성 확보를 위해서는 소프트웨어 차원의 신뢰성 향상 기법이 요구된다. 소프트웨어의 신뢰성 향상기법으로는 고장허용기법과 고장감내기법이 이용될 수 있으며, 이를 통하여 시스템 전체의 신뢰성 및 안전성을 향상시킬 수 있다.

열차제어 시스템 소프트웨어는 실시간성, 안전성, 신뢰성을 요구하는 고난이도의 분산 임베디드 소프트웨어라고 볼 수 있다. 임베디드 소프트웨어의 실시간성과 신뢰성을 분석하고 평가하는 연구는 기존에 많은 결과들이 발표되었으나, 소프트웨어의 안전성 평가에 대해서는 거의 연구가 이루어지지 않았다. 특히, 안전성 평가보다는 설계 단계에서 안전성 활동을 수행하는 것이 일반적으로 수행된 기법이었다. 본 연구에서는 설계 단계에서 소프트웨어의 안전성을 분석하는 것과는 달리, 소프트웨어의 테스트단계에서 사용할 수 있는 안전성 평가 기법을 제안하였다.

3. 소프트웨어 안전성 관련 국제표준

소프트웨어 안전성과 관련된 국제 표준들은 유럽을 중심으로 제정된 IEC61508-3[7], IEC62279[8]와 미국 국방부에서 제정한 DO-178B[9] 가 있다. IEC61508은 일반전자장치들에 적용하기 위한 안전성 표준인 반면에, IEC62279는 열차분야의 소프트웨어 안전성에 관한 표준이며, EC6DO-178B 는 항공분야에서의 소프트웨어 안전성 보장을 위한 표준요건을 규정하고 있다. IEC62279는 IEC61508과 매우 유사하므로 이 절에서는 IEC61508과 DO-178B 두 표준의 특성을 살펴본다.

3.1 IEC61508-3 안전성 표준

IEC61508은 전자장치의 안전성과 관련된 표준으로서 총 7 개의 파트로 구성되어 있다. 본 절에서는 IEC61508 의 파트 중에서 소프트웨어의 안전성에 대해 설명하는 세 번째 파트(IEC61508-3)의 내용을 소개한다. IEC61508-3 은 소프트웨어를 개발함에 있어서 안전성과 관련된 요구조건들을 열거함으로써 소프트웨어 개발자들이 그것을 준수하도록 하고 있다. 이 요구사항들은 소프트웨어의 안전성을 인증하기 위한 요소로서 사용될 수 있다. 따라서 본 절에서는 IEC61508-3에서 말하는 요구사항에 대해 소개한다. IEC61508-3에서 정의한 소프트웨어 개발 주기마다 요구사항들이 존재한다. 본 연구에서는 소프트웨어를 테스트 할 수 있는 단계의 요구사항이 필요하다. IEC61508-3에서 정의한 소프트웨어 개발 주기 중에서 그에 해당하는 단계는 코드검증, 데이터 검증, 소프트웨어 모듈 테스트, 소프트웨어 통합 테스트, 소프트웨어의 안전성 요구사항 테스트이다.

소스코드의 검증은 정적인 방법을 이용해 소프트웨어 모듈 설계가 적합한지, 개발에 요구되는 코딩 표준의 준수 여부와 소프트웨어 안전성 평가 계획의 요구사항에 적합하게 개발되었는지를 확인하도록 규정하고 있다. 데이터의 검증시에는 데이터구조, 응용어플리케이션 데이터, 인터페이스에 대해 완전성, 호환성, 보안성등의 검증을 요구하고 있다. 소프트웨어 테스트 단계에서 적용할 것이 권고 되고 있는 소프트웨어 안전성평가 기법들은 체크리스트, 정적/동적분석, 원인/결과 다이어그램, 이벤트트리분석(ETA), 고장트리분석(FTA), 소프트웨어 에러효과 분석, 확률시험, 성능시험, 기능시험 등이 있다.

3.2 DO-178B 소프트웨어 안전성 표준

DO-178B 에서는 소프트웨어의 안전성 등급을 시스템 고장시에 야기되는 사고의 심각성에 따라 레벨 A부터 E 까지의 5단계로 나누고 있으며, 레벨 A는 소프트웨어의 고장으로 인하여 항공기 제어시스템의 최대고장으로 이어지는 경우로 최고의 안전성을 요구하는 수준을 말한다. DO-178B 는 개발 단계를 소프트웨어 요구사항 분석, 소프트웨어 설계, 소프트웨어 개발, 소프트웨어 통합 등 4 단계로 나눈다.

DO-178B 의 소프트웨어 요구사항 분석의 결과는 높은 수준의 요구사항이다. DO-178B에서 정의한 높은 수준의 요구사항은 시스템 요구사항을 분석함으로써 얻어지는 것으로 기능적이고 운영적인 명세, 시간 및 메모리의 제약, 하드웨어와 소프트웨어의 인터페이스, 오류 발견과 안전성 감시 요구사항 등을

포함한다. DO-178B 의 소프트웨어 설계의 결과는 낮은 수준의 요구사항과 소프트웨어 구조이다. DO-178B에서 정의한 낮은 수준의 요구사항은 DO-178B에서 정의한 높은 수준의 요구사항을 분석함으로써 얻어진다. 그 내용으로는 입출력에 관한 설명, 데이터와 제어흐름, 자원 제약, 스케쥴링과 통신 방법 등이 포함된다. 소프트웨어 소스 코드에서 확인해야 할 안전성 관련 항목으로는 소프트웨어 구조의 준수여부, 검증가능성, 추적가능성 등을 포함하고 있다. 소프트웨어 테스트단계는 소프트웨어의 모든 요구사항에 대한 만족도 분석을 하는 소프트웨어 요구사항 커버리지 분석과 구현된 요구사항에 대한 만족도 분석을 하는 소프트웨어 구조 커버리지 분석으로 이루어진다.

4. 열차제어시스템 소프트웨어 안전성 평가 요구조건

열차제어 시스템 소프트웨어의 안전성을 확보하기 위해서는 IEC61508과 DO-178B에서 규정한 것처럼 소프트웨어 개발의 전 과정에 걸쳐서 소프트웨어의 품질관리가 이루어져야 한다. 여기에는 문서화, 개발프로세스별 요구사항, 품질관리체제 등에 있어서 철저한 관리를 요구한다. 열차제어시스템 소프트웨어의 안전성 인증작업은 이들 국제 표준에서 요구하는 요건들에 대한 만족도를 피인증기관에서 제출한 근거 서류를 검토함으로써 수행하게 된다. 그러나, 이러한 인증방법으로는 제작된 소프트웨어와 제출한 근거서류간의 일치 여부를 확인하기가 어렵다는 단점이 있다. 이러한 문제점을 극복하기 위해서 인증기관에서는 피인증 소프트웨어의 소스코드와 실행코드, 그리고 타겟시스템을 이용하여 안전성 활동의 결과를 검증하는 과정이 필요하다.

인증기관에서 필요로 하는 소프트웨어 안전성 평가도구는 자동으로 소프트웨어를 평가할 수 있어야 하며, 소프트웨어 개발 이후에 적용할 수 있는 기법으로 구성되어야 한다. 또한, 국제 표준에서 요구하는 평가 항목들 중에서 인증평가자의 판단하에 선택적으로 적용 가능하도록 구현되어야 한다.

따라서, 본 연구에서 제안하는 소프트웨어 안전성 평가 방법으로는 소프트웨어의 동적테스트 형태를 갖는 것이 바람직하다. 동적 테스트는 소프트웨어의 최종 실행 환경과 동일하게 구성하고 수행하거나, 그와 유사한 환경에서 수행하도록 한다. 소프트웨어의 안전성은 소프트웨어 개발 전 과정을 통하여 구현된 것이나, 인증시에는 최종제품을 이용하여 전과정의 결과물들을 검증할 수 있도록 해야 한다. 검증시에 가장 비중을 두어야 하는 것은 시스템의 기능안전성이 만족되고 있는가이다.

개발하는 안전성 평가도구는 소프트웨어 개발 주기에서 파생된 결과들을 입력으로 이용할 수 있도록 설계되어야 한다. 이는, 각 개발 단계의 결과를 확인할 수 있는 효과가 있을 뿐 만 아니라, HAZOP, FMECA, FTA 등과 같은 설계 단계의 주요 분석결과를 검증시에 이용할 수 있기 때문이다. 이를 위해서는 개발 단계에서 파생된 결과물들을 도구의 입력화 할 수 있는 명세언어를 이용하는 것이 바람직하다. 표준화된 명세언어를 이용하면, 다양한 입출력 도구를 이용할 수 있게 된다.

개발하는 안전성 평가도구는 평가항목을 재구성 할 수 있어야 한다. 국제표준에서 요구하는 평가항목이 다양하기 때문에, 모든 평가항목을 검증하는 것이 불가능 할 뿐만 아니라, 피인증기관에서 제출하는 문서로 검증이 가능한 경우에는 굳이 검증을 다시 수행할 필요가 없기 때문이다.

5. 열차제어시스템 소프트웨어 안전성 평가도구

기존 소프트웨어 안전성 평가에 관한 연구들은 평가의 관점이 아닌, 개발초기에 적용이 가능한 수동적 분석방법론들이 주류를 이루고 있으며, 이들을 종합한 결과, 기존의 연구결과나 판매되고 있는 안전성/신뢰성 분석도구들은 안전성평가용으로 활용되기에는 부족한 점이 많다고 판단된다. 기존의 안전성 분석기법들의 적용사례들을 볼 때, 대부분이 설계 및 개발과정에 집중되어 있으며, 인증 시에는 자동화된 도구를 이용하기 보다는 개발업체에서 제공하는 안전성활동 데이터를 이용하여 인증 및 평가가 이루어지고 있다. 또한, 설계 및 개발과정에서 이용하고 있는 안전성 분석 방법들도 대부분 개발자들의 전문성에 의존해야 하는 수동적인 과정으로 이루어져 있다. 최근 많이 이용되고 있는 모델기반분석기법은 모델의 정확성을 증명하는 것이 근본적인 문제로 남아있게 되어, 실제 안전성을 보장하기에는 부족한 점이 있다고 하겠다.

따라서 본 연구에서는 이들 결과를 종합하여 열차제어시스템 소프트웨어 안전성 평가기법으로 모델

기반분석이 아닌, 대상시스템을 직접 이용한 평가방법을 제안하고자 한다. 제안하는 방법은 인증단계에서 사용하게 되는 자동화된 소프트웨어 안전성 평가도구로 IEC62279에서 정의한 인증평가 기법 중에서 정적분석 및 동적분석을 중심으로 평가하고자 한다. 다음의 그림은 제안하는 안전성평가 기법의 구성을 보여준다.

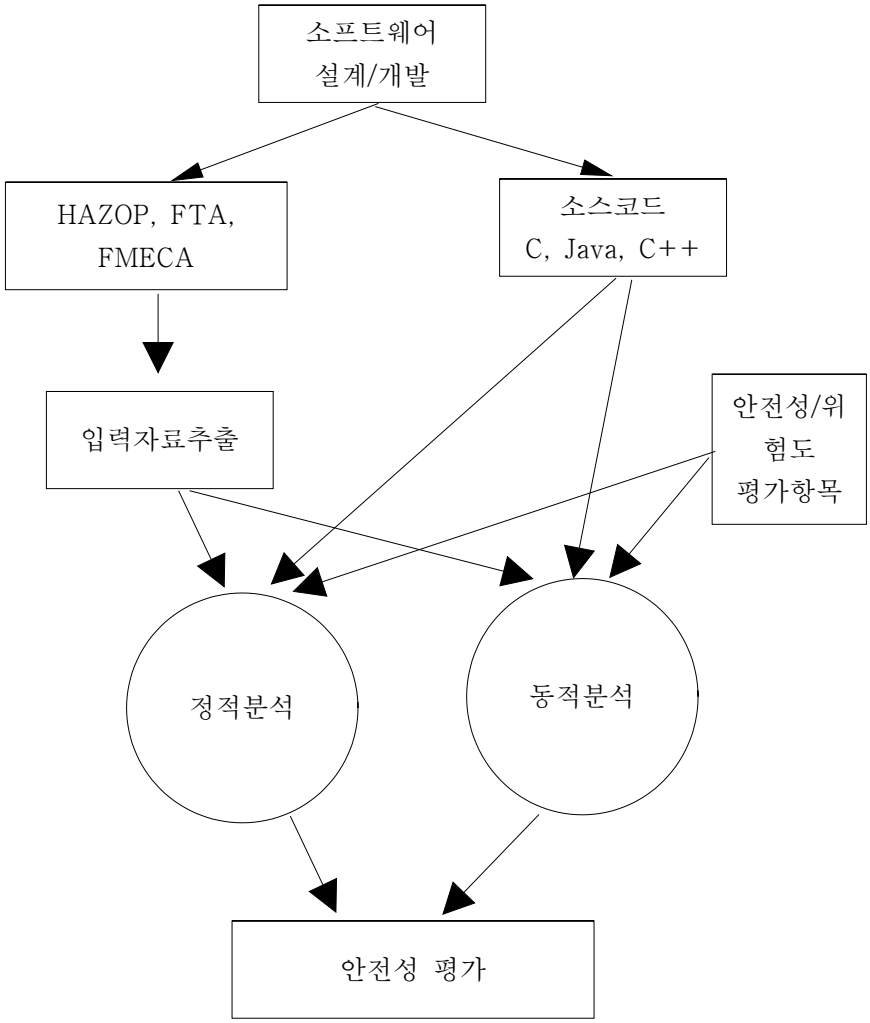

그림 1. 제안하는 안전성 평가 기법의 구성

인증단계에서 이용하는 것을 목적으로 하는 도구이므로, 소프트웨어 개발 동안에 이루어진 안전성활동 결과물을 입력으로 하고, 그 결과물 중에서 안전성 요구조건들을 추출한다. 소스코드를 이용하여 정적분석을 수행하여, 위험도 평가를 위한 기본변수들을 측정한다. 기본 변수는 일반적으로 사용되고 있는 복잡도 측정항목들을 사용할 수 있다. 제출된 FMEA, ETA, FTA 분석자료로부터 추출된 추가 시험항목들을 동적시험엔진의 입력으로 이용한다. 동적분석에서는 주요 해저드 발생빈도 측정을 주로 수행하며, 이를 이용하여 소프트웨어의 위험도를 평가한다.

실제 이와 같은 안전성 평가 자동화 도구의 구현은 기존의 임베디드 소프트웨어 시험도구를 확장하는 형태로 가능하다. 기존 시험도구의 구조는 다음과 같다.

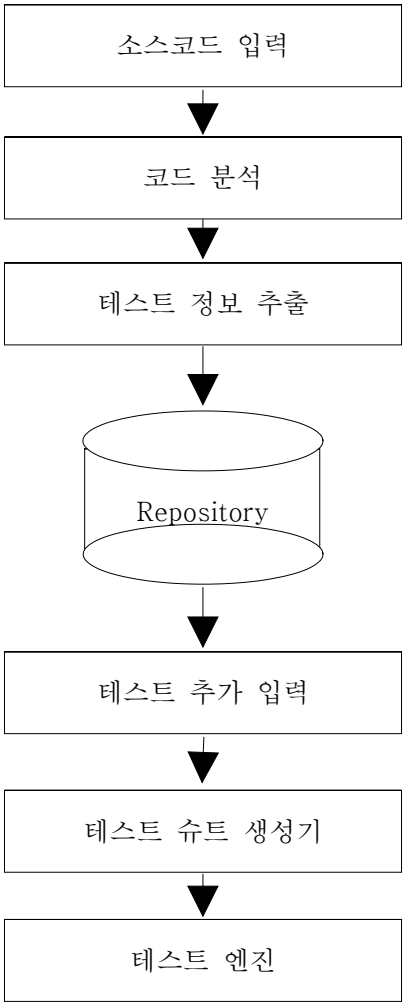

그림 2. 기존의 임베디드 소프트웨어 시험 도구

기존의 임베디드 소프트웨어 시험도구는 소프트웨어의 시험전문 도구로 안전성 평가 기능이 전혀 없는 상태이다. 따라서 이러한 도구에 본 연구에서 제안하는 안전성 평가기능을 다음과 같이 추가하면. 자동화된 안전성 평가도구로 활용할 수 있게 된다.

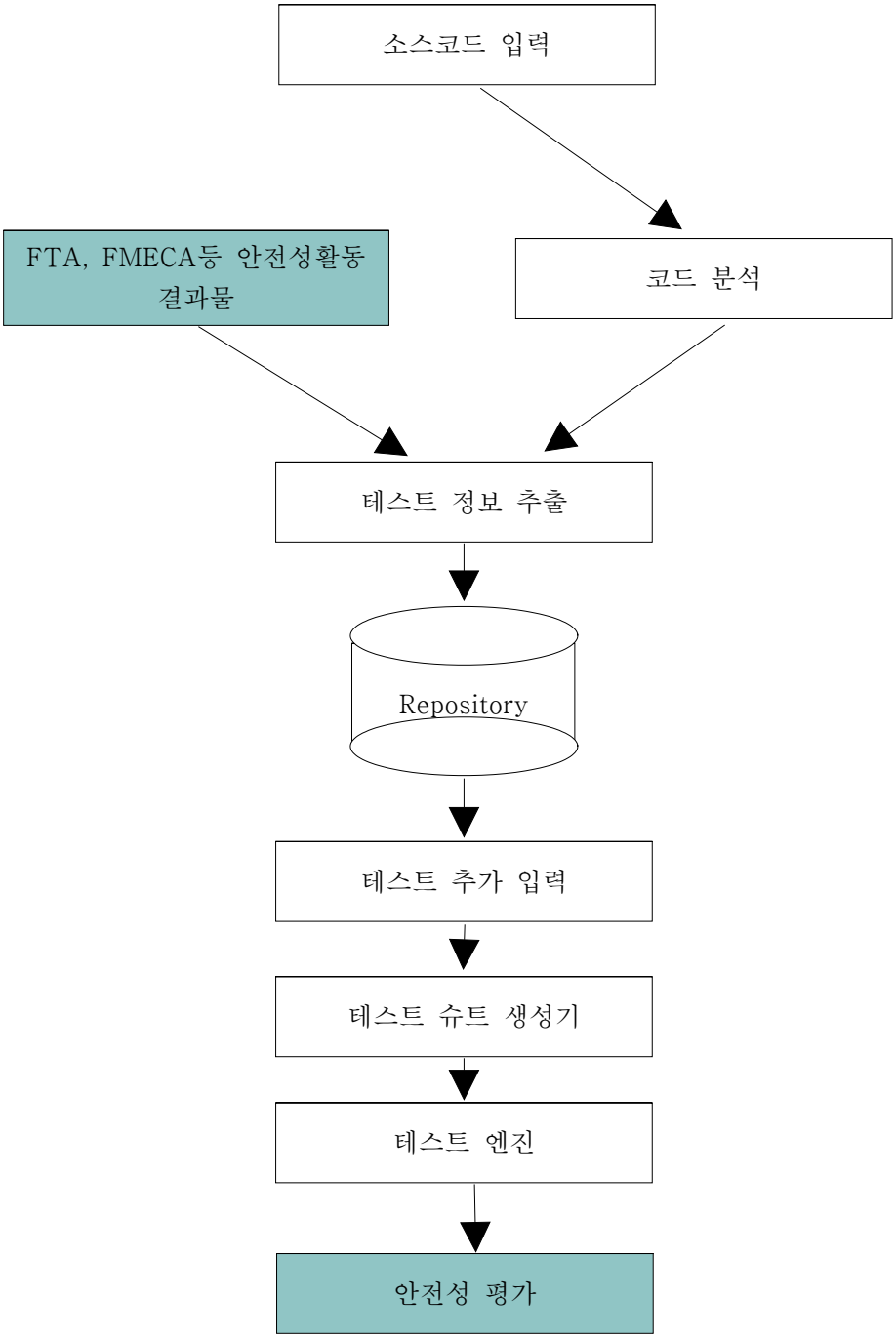

그림 3. 제안된 안전성 평가 도구의 구조

이와 같이 대상 시스템에 평가하고자 하는 소프트웨어를 직접 탑재하여 안전성을 평가하므로 모델을 이용하는 경우보다 실질적인 평가가 가능하다는 장점이 있다. 이러한 동적평가기법에서는 FTA나 FMECA등의 소프트웨어 개발과정에서 수행한 안전성 활동의 결과들을 테스트 슈트 생성에 얼마나 정확히 반영할 수 있는가에 달려 있으므로, 향후 도구의 실제 개발에 있어서는 FTA, FMECA등의 결과를 자동으로 테스트 슈트로 변환할 수 있는 방법에 대한 연구가 선행 되어야 할 것이다.

6. 결론

본 논문에서는 열차제어 시스템소프트웨어의 안전성 평가도구를 제안하였다. 제안한 평가도구는 기존의 자동화된 소프트웨어 테스트 도구를 확장하는 형태를 가지고 있으며, 소프트웨어 개발주기에서 파생된 안전성 활동의 결과들을 입력으로 이용하여 표준에서 요구하는 평가항목들을 동적테스트 형태

로 수행하도록 한다. 이를 위하여 열차제어 시스템 소프트웨어를 분석하였으며, 소프트웨어 안전성 관련 국제표준들에 대해서 조사/분석 하였다. IEC61508 과 DO-178B 를 통해 열차제어 시스템 소프트웨어의 안전성 관련 요구사항들을 정리했다.

제안된 구조를 갖는 임베디드 소프트웨어 시험도구가 개발된다면 열차제어 시스템의 소프트웨어 안전성을 평가하는데 큰 도움을 줄 수 있을 것으로 기대된다.

7. 참고문헌

1. M. Matsumo, "The revolution of train control system in Japan", Autonomous Decentralized Systems, 2005. ISADS Proceedings, 2005, pp.599 - 606

2. H.W. Lawson et al, "Twenty years of safe train control in Sweden", Engineering of Computer Based Systems, Proceedings. Eighth Annual IEEE International Conference and Workshop on the 2001 Page(s):289 - 297

3. Younju Oh et al., "Software safety analysis of function block diagrams using fault trees", Reliability Engineering & System Safety, Vol. 88, No. 3., pp. 215-228., 2005

4. W. Weber et al.. "Enhancing software safety by fault trees: experiences from an application to flight critical software.", Reliability Engineering & System Safety, Vol. 89, No. 1., pp. 57-70, 2005

5. Ian B. Pirie, "Software ?How do we know it is safe?", in Railroad Conference, 1999. Proceedings of the 1999 ASME/IEEE Joint, pp.122-129, 1999

6. Robyn R. Lutz and Robert M. Woodhouse, “Bi-directional Analysis for Certification of Safety-Critical Software”, in proceedings of 1st International Software Assurance Certification Conference, Dulles, Virginia, February 1999

7. International Electrotechnical Commission (IEC) (1999), "61508 - Functional Safety of Electrical / Electronic / Programmable Electronic Safety-Related Systems"

8. International Electrotechnical Commission (IEC) (2002) "62279 Railway Applications - Communications, Signalling &. Processing Systems, Software for Railway Control & Protection"

9. RTCA DO-178B/ED-12B (1992), "Software Considerations in Airborne Systems and Equipment Certification"