S/W 안전성을 위한 분석기법 조합과 개발 프로세스 평가에 대한 연구

A Study on the Analytic Technique Combination and Evaluation of Development Process for Software Safety

이영수* 안진 ** 하승태*** 조우식**** 한찬희*****

Young-Soo Lee Jin Ahn Seung-Tea Ha Woo-Sik Cho Chan-Hee Han

ABSTRACT

The goal of this thesis is to support safety and reliability characteristics of software intensive critical systems. The verification method developed is innovative from current state of the art in what concerns the verification viewpoint adopted: focusing on software faults, and not, like many other approaches purely on fulfilling functional requirements. As a first step and based on a number of well defined criteria a comparison was made of available literature in the area of static non formal non probabilistic software fault removal techniques. But, None of the techniques evaluated fulfilled all criteria set in isolation. Therefore a new technique was developed based on a combination of two existing techniques: the FMEA and FTA. These two techniques complement each other very well. It is possible to integrate both techniques with commonly used techniques at system level. The resulting new technique can be shown to combine nearly all aspects of existing fault removal techniques.

---

1. 서론

열차제어시스템의 운영에 있어 시스템의 오작동은 사소한 사고에서부터 돌이킬 수 없는 재난의 결과를 가져올 수 있으며, 사실상 지금도 다양하고 많은 사례의 사고를 불러일으키고 있다. 이는 물론 소프트웨어의 오류만이 아니라 하드웨어적인 오류 및 오작동까지 포함하고 있지만, 마이크로프로세스의 발전으로 점차 그 비중이 높아져만 가는 소프트웨어의 오류를 줄여야만 전체 시스템의 신뢰성 및 안전성이 확립될 수 있으리라 확신한다.

소프트웨어의 결함은 위험원 분석과 평가를 통해 도출 및 분석이 가능하며, 이는 전 수명주기를 통해 위험상태의 파악과 파악된 위험원이 허용 가능한 범위에 있는가의 여부 확인 그리고 그에 대한 대책을 마련하는데 그 목적이 있다. 소프트웨어의 신뢰성 및 안전성을 위한 위험원 분석기법은 오래전부터 제시되어 왔으나 이러한 활동과 프로세스에 대한 평가가 이루어지지 않아 그 효율성이 모호하였으며, 그 척도 또한 불확실한 실정이었다.

---

* 경봉기술(주) 책임연구원

** 경봉기술(주) 책임연구원

*** 경봉기술(주) 선임연구원

**** 경봉기술(주) 선임연구원

***** 경봉기술(주) 주임연구원

본 논문에서는 열차제어시스템 소프트웨어 개발에 적합하고 효율적인 위험원 분석기법에 대해 이미 여러 문헌을 통해 확인된 사항에 대해 언급할 것이며, 이러한 활동을 평가할 수 있는 프로세스 평가기법을 ISO/IEC 15504를 적용하여 제안하였다.

2. 문제점 제시

소프트웨어의 일반적인 생명주기상의 테스트만으로는 내재된 잠재적인 모든 위험원을 찾아 처리하기가 어렵다.

소프트웨어의 특성상 시간 경과에 따른 발생 가능한 위험원은 충분한 테스트를 거쳤다 하더라도 위험원에 대한 대책을 마련하지는 못한다.

점차 소프트웨어 구조의 복잡성이 높아져가는 상황에서 분석기법과 개발 프로세스의 접목이 반드시 필요하다.

안전성과 신뢰성은 장치와 그 장치를 작동하는 사람에 내재된 시스템 특성이다. 장치 자체의 안전성과 신뢰성은 충분한 테스트와 소프트웨어나 하드웨어 가운데 어느 하나의 컴포넌트가 제대로 작동하지 않을 때 사용자 및 기타의 인명을 보호하는 고장 안전 메커니즘(Fail-Safe)을 바탕으로 한다.
신뢰성과 안전성은 서로 연관되어 있지만, 동일한 개념은 아니다. 신뢰성(Reliability)은 시스템이 어떤 종류의 고장도 없이 실행될 수 있는 확률적 개념이고, 안전성(Safety)은 치명적인 고장없이 시스템이 실행된다는 의미이다. 그러므로 소프트웨어 안전성 및 신뢰성은 소프트웨어 결함에 의해 유발되는 소프트웨어 고장과 관계가 있다.
소프트웨어 결함 예방, 결함 허용, 결함 제거 및 결함 예측은 소프트웨어의 안전성 및 신뢰성 확보를 위한 것으로, 핵심 시스템의 소프트웨어를 위한 사용과 구현 그리고 검증에 활용되는 기법이다. 소프트웨어 제품을 개발할 때 이들 기법을 활용하려면, 이들 기법과 개발 프로세스, 소프트웨어 개발을 위한 방법과 기법, 그리고 서로 다른 제품 아키텍처 사이의 관계를 먼저 확립해야 한다.

3. 연구방향

본 연구에서는 소프트웨어의 개발에 있어 발생 가능한 결함을 제거할 수 있는 위험원분석기법을 제시하려 한다. 또한 위험원 분석기법 이후에 국제규격에서 규정하고 있는 프로세스와의 적용성에 대해 논의할 것이다.

우선적으로 용어에 대해 정리하면 다음과 같다.

- 결함(Fault)은 “특정 운영 조건에서 이상으로 이어질 수 있는 시스템의 불완전성”으로 정의된다. 즉, 시스템에 일시적 또는 영구적으로 자리하고 있는 하드웨어나 소프트웨어의 오류를 유발하는 원인으로 풀이할 수 있다.
- 오류(Error)는 “계산, 관찰, 측정 값 또는 지정된 정확한 값/조건과의 차이”로 정의된다. [IEC61508], [EN50128], [ECSS]
- 고장(Failure)은 “필요기능을 수행할 수 있는 능력이 종식된 것”으로 정의된다. [ECSS], [IEC61508]. 또한 EN50128에서는 “소프트웨어의 고장은 오류의 발현이다”라고 정의하고 있다.
- 결함제거 (Fault removal)는 소프트웨어 결함을 제거하는 방법으로 관찰된 고장분석, 원인파악, 결함제거를 수행하게 된다.
- 결함예측 (Fault forecasting)은 현재 존재하는 결함의 수, 발생가능한 결함, 결함의 결과를 추정하는 방법으로 정의된다.

설계단계에서 적절한 결함 허용기법의 채택을 계획하려면, 요구 기준 설정 과정 중에 결함제거 기법을

채택해, 결함 허용기법이 필요한 곳이 어디이고 어떤 것이 필요하며 어떻게 적용할지 파악할 필요가 있다. 검증 프로세스에서는 결함제거 기법을 활용하여 100% 테스트가 불가능한 부분을 보완할 필요가 있다.
개발 단계부터 조기에 결함발생을 제거함으로써, 운영 단계에서 필요한 결함제거 기법의 활용 수준을 감소시킬 수 있다.

3.1 소프트웨어 결함제거 단계

소프트웨어 결함은 고장의 원인이므로, 소프트웨어 결함을 제거하면 시스템의 안전성과 신뢰성이 직접적으로 개선된다. 특히 본 논문에서는 결함이 발생하여 차후에 소프트웨어 고장을 유발해 치명적인 결과를 가져올 수 있는 개발 단계의 소프트웨어 결함제거에 중점을 둔다. 여러 문헌에서는 오류를 일찍 찾아낼수록 오류를 수정하는 비용이 크게 감소한다고 밝히고 있다.
첫 번째 단계는 소프트웨어 제품에서 피해야 할 핵심 소프트웨어 고장모드를 파악하는 것이다. 이후 결함 제거 단계가 이어지는데 순차적으로 다음과 같다.

- 결함 감지 : 결함을 발견하거나 발견하도록 설계된 단계 (결함이 발생했음을 결정하는 단계)
- 결함 분리 : 결함의 출처 또는 위치를 파악하기 위한 단계
- 결함 복구 : 결함을 제거하거나 결함 발생을 피하기 위한 단계 (결함을 제거하거나 이의 통제 도는 허용을 위한 수단을 제공)

위험원분석기법은 위의 단계를 모두 포괄해야 하며, 소프트웨어 개발 프로세스의 최종 운영 단계서만 적용해서는 안된다. 이 모든 단계를 각종 소프트웨어 개발 단계의 활동에 완전히 통합시켜, 소프트웨어 결함을 가능하면 개발 초기 단계에 제거해야 한다.

3.2 기법의 분석

다음의 표는 다양한 위험원분석 기법에 대한 내용을 정리한 내용이다.

분석 기법	설명
알고리듬 분석 (algorithm analysis)	알고리듬의 정확성, 적절성, 안정성을 점검하고, 알고리듬이 모든 정확성, 타이밍, 사이징 기준에 부합하는지 점검한다. 알고리듬 분석은 알고리듬, 등식, 수학 공식 또는 표현의 올바른 구현 여부를 시스템 또는 소프트웨어 기준에 대비하여 확인한다.
원인결과분석법/도식 (cause consequence analysis/diagrams)	기본 이벤트의 조합에 따른 결과로 시스템에 발생할 수 있는 이벤트 순서의 도식적 모델링
공통원인이상분석 (common cause failure analysis)	동시에 예비 부분에서 동일한 이상이 발생함으로써, 예비 시스템 구비의 혜택을 훼손시킬 수 있는, 예비 시스템 또는 서브시스템의 잠재 이상 파악
제어흐름분석/도식 (control flow analysis/diagrams)	예정 제어 흐름에 문제가 없는지 점검(예, 도달 불능 또는 부정확한 디자인 또는 코드 요소). 대규모 프로젝트인 경우에 이들 도식은 프로그램 제어 흐름 이해에 유용하다.
데이터흐름분석 (data flow analysis)	프로그램 실행 시에 프로그램 변수가 초기화, 변형, 또는 조회될 때, 프로그램 변수의 행동 점검.
ETA(event tree analysis)	개시 이벤트 이후 시스템에서 발생하는 이벤트 순서를 도식적으로 모델링하여, 그에 따른 결과가 얼마나 심각한지 보여 주는 방법
FMEA(failure modes and effect analysis)	FMEA와 FMECA는 제품의 잠재적 장애 모드, 이들 장애의 영향, 그리고 그 중요성을 체계적으로 파악하는 절차이다. SEEA는 소프트웨어 디자인 컴포넌트를 평가하여, 소프트웨어 이상 모드가 다른 디자인 요소, 인터페이스 컴포넌트, 또는 소프트웨어 컴포넌트의 기능, 특히 핵심적인 기능에 미치는 파급효과를 파악한다.
FTA(fault tree analysis)	독자적으로 또는 다른 것과 함께 제품의 지정 상태(장애, 불안전한 조건 등)로 이어지게 하는 원인(내적 또는 외적)의 파악을 위한 체계적 접근 방법
HA(hazard analysis)	시스템의 위해 요소를 파악하고 평가하며, 다음에 그 위해 요소를 배제하거나 "허용

	수준"으로 리스크를 감소시키는 변경 권고를 제기하는 프로세스.
HAZOP(hazard and operability analysis)	컴퓨터 시스템 컴포넌트 섹션과 컴퓨터 시스템의 운영 상태를 체계적으로 검사함으로써, 제어 시스템에서 잠재적 위해 상황을 유발할 수 있는 이상 모드를 파악하는 방법.
메트릭(metrics)	소프트웨어 자체의 특징을 감안하여, 프로그램 특성의 계량적 예측
페트리 네트(Petri Nets)	시스템 행동의 관련 측면을 모델링하고 분석과 재디자인을 통해 안전성 및 운영 기준을 평가하고 개선하는 그래픽 기법.
안전성특징분석/최악의경우분석(safety properties analysis/worst-case analysis)	타이밍, 정확도, 성능을 포함하여, 비기능적 안전성 특징의 최악의 경우 조건 분석.
SCA(sneak circuit analysis)	바람직하지 않은 프로그램 기능을 유발하거나 원하는 기능을 저해하는 예상치 못한 경로 또는 논리 흐름 파악. 스니크는 전기, 소프트웨어, 통합 시스템에 우연히 통합되는, 잠재성 디자인 조건 또는 디자인 결함을 의미한다. 컴포넌트 이상에 의해 유발되지 않는다.

4. 연구내용

본 연구에서 중점적으로 다룬 내용은 위험원 분석활동을 평가하기 위한 프로세스를 제안하는 것이다. 우선적으로 위험원 분석활동의 적합한 기법들의 조합에 대해 언급할 것이며, 이러한 분석활동을 평가하기 위한 프로세스를 제시할 것이다.

소프트웨어의 안전성 효과를 극대화 할 수 있는 다양한 위험원분석기법이 존재하나 모든 기법을 적용할 필요는 없다. 개발해야 할 소프트웨어의 특성과 적합성을 고려하여 선택하여야 하지만, 각 기법들이 가지고 있는 특이성으로 인해 단 하나의 기법을 적용할 경우 완벽한 분석이 이루어지기 어렵기에 두 가지 이상의 기법을 적용할 것을 본 논문에서는 권고한다.
여러 가지 위험원분석기법의 조합 중에 FMEA와 FTA의 조합은 이미 여러 연구문헌에서 제시된 바와 같이 FTA기법은 다른 기법과 효과적으로 조합할 수 있으며, 아래 설명과 같이 상호보완적인 특성이 있기에 선택되었다.
FMEA는 고장의 중요성과 심각성 파악에 중점을 두고, FTA는 결함의 원인을 파악하는데 중점을 둔다. 또한 FMEA는 전적으로 상향식 접근 방식이며, FTA는 이를 보완할 수 있는 하향식 접근 방식이다.
FMEA와 FTA의 기술적인 내용은 여러 문헌에서 다루고 있기에 본 논문에서는 기술하지 않았으며, 단지 두 기법의 조합 결과에 대해 다음과 같이 확인하여야 한다.

두 기법의 조합은 다음의 요구사항을 충족시켜야 한다.

- 반복성
  FMEA와 FTA 기법의 조합을 활용하여 동일한 결과를 확보하기 위해 소프트웨어 및 조합 활용에 관한 세부적인 단계별 적용 가이드라인을 규정한다.
- 정확성
  기법의 조합시에 단계별로 활용할 입력 세부 사항과 확보할 예상 결과는 의견이나 특정 결과에 우호적인 편향적 결과를 바탕으로 하지 않는다. 즉 사실에 중점을 둔다.
- 가용성
  단계별로 그 방법의 활용과 관련한 제약조건을 세부적으로 규정한다.
- 신뢰성
  결과에 영향을 미칠 수 있는 단계별 변동 부분을 규정한다.
- 감당성
  FMEA 테이블과 FTA 계통도는 범위가 크고 복잡하다. 제한적인 고장 또는 결함 분류 세트를 활용하고 서로 다른 소프트웨어 생명주기 단계와 소프트웨어 구조의 다양한 컴포넌트에 따라 단계별 접근 방식을 규정한다.

- 결과의 의미성
  결과가 이해할 수 있고 체계적이고 유용한 결과를 제공해야 하며, 소프트웨어에서 제거할 결함을 명확히 표시해야 한다.
- 지시성
  두 기법의 결과는 대상 소프트웨어의 어느 부분이 개선되어야 하는지 명확히 보여주어야 한다.
- 기법의 이해성
  두 기법의 조합으로 인한 소프트웨어 고장 및 결함에 적용하는 방법에 관한 가이드라인을 제공해야 하며, 모든 결함제거 단계에 대한 가이드라인이 명확히 규정되어야 한다.

4.1 FMEA (Failure Mode Effect Analysis)의 예

FMEA를 수행할 때는 소프트웨어의 기능에 대한 잠재적 고장 모드를 파악하여 소프트웨어 고장을 분석한다. 다음은 CTC 소프트웨어 하나의 기능에 대한 FMEA분석 도표 일부이다.

번호	고장모드	원인	영향	관찰 가능한 증상	예방 및 처리방법
1	선로전환기 정보가 현장 선로전환기 정보와 불일치	현장과의 I/O 불일치	선로전환기 변환이 되지 않음.	선로전환기 갱신안됨 선로전환기화면표시와 달리 열차운행	현장의 하드웨어 또는 data확인 현장과의 DB동기화
2	열차접근에 대한 전송처리 오류	열차번호 미수신 궤도 접근 분류 불일치 실적 DB참조 오류 열차스케줄 참조 오류	열차접근 정보 확인 불가	운행중인 열차 열번 없이 운행 스케줄에 없는 열차 운행	열차접근처리 로직 확인 및 수정 열차스케줄서버 확인 및 수정 통신회로 이상 유무 확인

4.2 FTA (Fault Tree Analysis)의 예

FTA의 문서화는 일반적인 FTA원인 계통도를 제공하는 직접적인 방법이 있고, 또는 표 형식으로 모든 권고사항을 제시하는 방법을 사용할 수 있다.

FTA에서 파악된 결함 모드의 원인일 수 있는 기본 이벤트에 대하여, 이의 제거 방법에 대해 제시해야 한다. 제거 방법은 결함의 시정을 위한 정확한 소프트웨어 아이템(컴포넌트, 코드, 모듈, 변수 등)을 대상으로 해야 한다.

4.3 분석의 결과

FMEA와 FTA 조합의 결과를 평가하여 다음의 사항을 파악한다.

- 소프트웨어의 신뢰성 및 성능 특성에 영향을 주는 요소
- 하나 이상의 소프트웨어 컴포넌트에 영향을 주는 공통 이벤트
- 기타 분석 또는 프로세스에서 수립한 가정과 설계를 위반하지 않았으며 관련 컴포넌트가 고장을 유발할 수 없다는 것에 대한 증명
- 시스템 고장을 유발할 수 있는 이벤트의 파악
- 핵심 컴포넌트 및 이상 메커니즘에 대한 권고 사항 및 복구/유지관리 전략을 위한 의견

이상과 같은 분석기법의 조합으로 인한 활동을 하였지만, 개발 프로세스의 품질이 소프트웨어의 안전성 및 신뢰성 특성에 어떻게 영향을 미치는지를 확인할 수 없었다. 그래서 소프트웨어 개발 프로세스 평가에 대한 표준문서인 ISO/IEC15504에 규정된 사항에 대해 검토하였다.

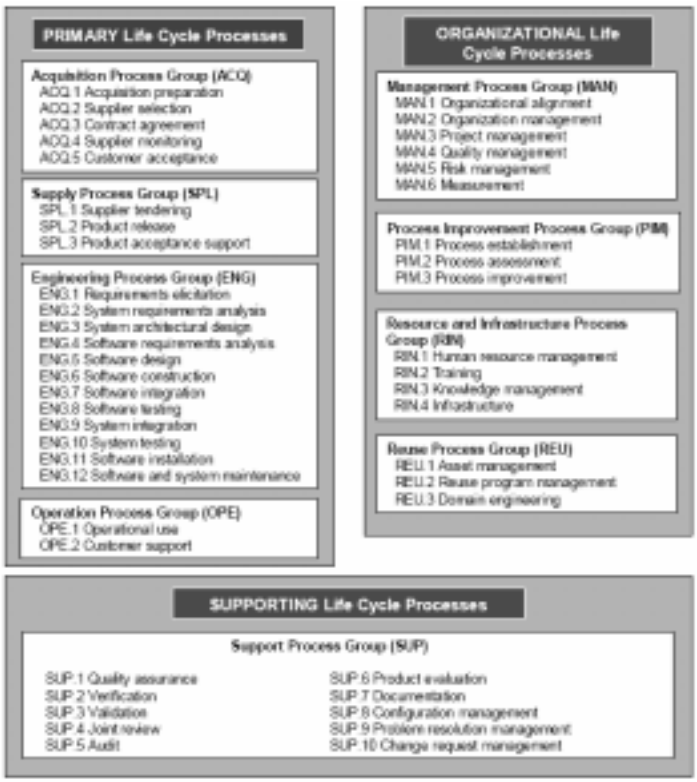
4.4 프로세스 평가 모델 도입(ISO/IEC15504)

4.4.1 ISO/IEC15504

ISO/IEC15504는 현재의 소프트웨어 개발 프로세스에 대해 평가하여 프로세스의 부족한 부분을 파악하고 그에 대한 지속적인 관리와 개선이 이루어지도록 방향을 제시해 주는 방향자의 역할을 한다. 안전성 활동은 개별적이거나 독립적인 활동이 아니라, 전체 생명주기에서 포함되어 설계, 개발, 품질과 유기적인 관계를 가진 활동이기에 소프트웨어 개발 프로세스를 평가하여 안전성 활동이 적합하게 이루어졌는지 평가할 필요가 있다.

ISO/IEC15504에서 프로세스를 평가하기 위해 48개의 프로세스를 정의하고 있으며, 48개의 프로세스는 9개의 그룹과 3개의 큰 Life Cycle로 나뉘어져 있어 평가 받고자 하는 프로세스를 선택하기 용이하게 구성되어 있다.

ISO/IEC15504의 각 프로세스는 Base Practices와 Work Product를 가지고 있으며, Work Product의 요구사항을 만족하는 평가를 해야 한다.



또한 ISO/IEC15504는 48개의 프로세스 중 평가 받고자 하는 프로세스를 평가하여 이에 대한 능력을 측정할 수 있는 6단계의 레벨을 제시하고 있다.

- Level 0 : Incomplete Process 불완전한 프로세스
- Level 1 : Performed process 수행된 프로세스
- Level 2 : Managed process 관리된 프로세스
- Level 3 : Established process 수립된 프로세스
- Level 4 : Predictable process 예측 가능한 프로세스
- Level 5 : Optimizing process 최적화한 프로세스

4.4.2 위험원 분석활동의 평가

본 연구에서는 위험원 도출 및 분석활동이 전체 개발 프로세스에서 미치는 영향을 평가할 수 있는 방법을 제시하려 하며, 구체적으로 위험원 분석활동의 요구사항을 ISO/IEC15504의 프로세스를 이용하여 수립할 수 있는 방안을 찾고자 수행되었다.

ISO/IEC15504는 3개의 Life Cycle로 구성되어 있어 모든 프로세스는 일반적인 개발 Life Cycle과 상응되는 흐름을 가지고 있으며, 소프트웨어의 개발에 관련된 사항 뿐 아니라 조직구성과 소프트웨어의 품질에 관한 내용을 모두 포함하고 있다.

여기서 중점적으로 기술하고자 하는 내용은 FMEA와 FTA 활동 또는 전반적인 위험원 분석활동의 평가를 수행하기 위해 필요한 요구사항을 ISO/IEC15504의 48개 프로세스 중에서 도출하여 이에 대한 만족도를 측정할 수 있는 방법을 제시하는 것이다.

우선적으로 위험원 분석활동과 관련된 프로세스를 선정하여 Base Practices와 Work Product의 요구사항을 분석하였다. 위험원을 도출하고 분석하기 위해서는 개발하고자 하는 소프트웨어의 요구사항이 기본적으로 투입되어야 하며, 분석된 결과에 대해 설계되고 테스트하여 위험원에 대한 증명과 검증이 이루어질 것이다. 이상과 같은 활동 전반에 대해 ISO/IEC15504에서 이와 관련 프로세스를 분석한 결과 다음의 표와 같이 18개의 프로세스를 도출하였다.

관련성	Process ID	Process Name
★★☆☆☆	ENG.2	System requirement analysis
★★☆☆☆	ENG.3	System architecture design
★★★☆☆	ENG.4	Software requirement analysis
★★☆☆☆	ENG.5	Software design
★☆☆☆☆	ENG.6	Software construction
★★☆☆☆	ENG.7	Software integration
★★★★★	ENG.8	Software testing
★★☆☆☆	ENG.9	System integration
★★★☆☆	ENG.10	System testing
★★☆☆☆	ENG.12	System and software maintenance
★★★☆☆	SUP.1	Quality assurance
★★★★☆	SUP.2	Verification
★★★☆☆	SUP.3	Validation
★★☆☆☆	SUP.4	Joint Review
★★★★★	SUP.9	Problem resolution
★★☆☆☆	SUP.10	Change Request Management
★★★★★	MAN.5	Risk management
★★☆☆☆	MAN.6	Measurement

그러나 모든 프로세스에 대해 평가하여 만족한 수준의 평가레벨에 도달하기엔 여러 가지 요인으로 인해 매우 어렵기 때문에 관련성 측정치를 두어 관련성이 깊은 프로세스를 선택하여 시간과 비용을 줄일 수 있도록 하였다.

위의 위험원 분석활동과 관련성이 매우 깊은 관련성 5개의 별을 획득한 프로세스들은 개발 소프트웨어의 도출된 결함과 문제점에 대해 테스트를 수행해야 하며 지속적인 관리가 이루어져야 함을 나타내고

있다.
반면 관련성 정도는 다른 방향(조직구성, 테스트, 공급자 선정, 운영, 형상관리 등)의 프로세스와 병행하여 평가를 받게 될 경우 중복되는 경우가 발생할 수 있으며 이 경우엔 위의 표에서 제시된 관련성은 수정되며, 조직에서는 적합한 프로세스를 재선택해야 한다.

최종적으로 위에서 기술된 프로세스 능력을 측정하여 어느 레벨에 속하는지 확인을 해야 한다. 조직에서 소프트웨어를 개발하면서 도출된 위험원은 개발과정에서 제거되거나 비용과 시간상 제거하지 못하는 위험원은 지속적인 관리와 유지보수가 필요하게 된다. 그러기 위해서는 조직내에 이를 관리하기 위한 교육과 훈련 그리고 인적 자원배분이 이루어질 수 있는 기반이 마련되어야 한다.
그래서 본 연구에서는 위험원 분석활동의 모든 환경과 기반이 구축되어 지속적인 개선이 이루어져야 함을 주장하기에 프로세스 능력레벨은 Level 3의 Established process 이상 결정되어야 한다.
ISO/IEC15504에서 Level 3을 달성할 경우에 대해서 다음과 같이 정의하고 있다.

3.1 프로세스 정의 속성
- 적절한 tailoring 지침을 포함한 표준 프로세스에는 정의된 프로세스에 포함되어야 하는 기본적인 요소가 기술된다.
- 다른 프로세스와 함께 표준 프로세스의 순서와 상호작용이 결정된다.
- 프로세스를 수행하기 위해 요구되는 자격과 역할이 표준 프로세스의 부분으로서 식별된다.
- 프로세스를 수행하기 위해 요구되는 기반구조와 작업환경이 표준 프로세스의 일부분으로 식별된다.
- 프로세스의 효과성과 적절성을 감시하기 위한 적절한 방법이 결정된다.

3.2 프로세스의 전개 속성
- 적절하게 선택되거나 조정된 표준 프로세스에 기반하여 정의된 프로세스가 전개된다.
- 정의된 프로세스를 수행하기 위해 요구되는 역할, 책임과 권한이 할당되고 의사소통 된다.
- 정의된 프로세스를 수행하는 사람은 적절한 교육, 훈련, 경험에 대한 자격이 있어야 한다.
- 정의된 프로세스를 수행하기 위해 요구되는 자원과 정보가 할당되고 사용된다.
- 정의된 프로세스를 수행하기 위해 요구되는 기반구조와 작업환경이 사용, 관리, 유지된다.
- 프로세스의 상태를 이해하고, 프로세스의 적절성과 효과성을 확보하고, 지속적인 프로세스 개선이 이루어질 수 있는 부분을 평가하기 위하여 적절한 데이터가 수집되고 분석된다.

ISO/IEC15504를 이용한 소프트웨어 개발 프로세스의 평가는 FMEA와 FTA를 이용한 분석기법의 결과에 대해 다시한번 체계적인 프로세스 평가기법을 이용하여 확인하고자 하는 활동이며, 단편적인 하나의 프로세스가 아닌 유기적으로 관련된 여러 가지의 프로세스에 대해 평가하여 보다 나은 개발 활동이 이루어지도록 하여야 한다.

5. 결론

FMEA나 FTA 같은 전통적인 시스템 안전성 및 신뢰성 분석 기법을, 소프트웨어에 적용하여 동적 기법을 보완할 수 있다. FMEA와 FTA는 각기 상향식 및 하향식 기법으로, 여러 영역에서 시스템 분석에 널리 사용되고 있다. FMEA는 시스템의 소프트웨어 고장에 따른 잠재적 위험요소를 파악하고, 그 소프트웨어가 구현할 가장 핵심적인 기능을 파악하는데 도움이 된다. 또한 하위 레벨 소프트웨어 분해 및 구현을 조합하여 소프트웨어 행동을 분석하는데 도움이 되며, 이는 고장의 근본 원인을 파악하고 앞서 파악한 핵심 기능의 적절한 구현을 검증하는데 매우 유용하다. 이러한 정성적 안전성 및 신뢰성 검증 프로세스는, 분석 시에 가정된 것과 이후 운영 단계에서 나타날 고장 사항의 비교를 위한 토대로 활용될 수 있다.
소프트웨어 프로세스 심사를 통해 현재 진행하고 있거나 완료된 프로세스의 성숙도를 측정하는 것으로써,

ISO/IEC15504의 48개 프로세스 중 18개의 프로세스를 제시하였으며, 그 중 위험원의 제거와 관리를 위해 반드시 수행해야 할 프로세스로 다음과 같이 3개를 선정하였다.

- ENG.8 Software testing
- SUP.9 Problem Resolution
- MAN.5 Risk Management

소프트웨어 개발은 기술의 발전과 더불어 매우 복잡하고 규모가 커지고 있기에 그에 따른 오류 및 고장의 확률도 높아지고 있다. 그래서 다양한 개발 방법론과 위험원 분석기법의 적용으로 오류 및 고장의 원인을 제거하고 있다. 그러나 단편적인 방법론 또는 분석기법으로 모든 내재된 위험원을 도출하고 제거할 수 없다. 따라서 본 논문에서 제시한 조합형 분석기법과 같이 개발 방법론, 분석기법, 테스팅 기법, 프로세스 평가기술 등을 적합하게 조합하여 개발의 효율성과 안전성 신뢰성을 높일 수 있는 기술이 필요하다.

본 연구는 현재 국내 열차제어시스템의 개발에 적용하고 있는 다양한 안전성활동의 체계적인 프로세스를 확립할 수 있는 기반을 마련하고자 수행되었으며, 앞으로 안전성 활동 중 위험원 분석뿐만이 아니라 더 광범위한 테스트와 소프트웨어 유지보수에 이르는 프로세스에 대해 연구하여 안전성활동이 소프트웨어의 전반적인 품질활동과 유기적으로 맞물려 명확한 개발활동이 이루어질 수 있도록 연구되어야 할 것이다.

참고문헌

1.[ECSS] European cooperation for Space Standard, www.estec.esa.nl/ecss/, 13 April 1999
2.[EN50126] The specification and demonstration of dependability, reliability availability, maintainability and safety(RAMS).
3.[EN50128] Software for railway protection and control systems.
4.[EN50129] Safety related electironic systems for signalling.
5.[IEC61508] Functional safety : safety-related systems. IEC 1999.
6.[ISO15504] ISO/IEC Software Process Assessment-Part1 : Concepts and introductory guide.