ISSN 2234-4438
중소기업융합학회 논문지 제6권 제1호 pp. 7-15, 2016 DOI : http://dx.doi.org/10.22156/CS4SMB.2016.6.1.007

소프트웨어 개발 프로세스에서의 안전성 분석 및 관리 활동의 적용방안

김순겸, 홍장의*
충북대학교 전자정보대학 컴퓨터과학과

Application of Safety Analysis and Management in Software Development Process

Soon-Kyeom Kim, Jang-Eui Hong*
Department of Computer Science, Chungbuk National University

요 약 현대 사회에서 자동차, 철도, 항공우주, 원자력, 국방 등의 다양한 분야에서 대부분의 장치들이 소프트웨어를 내장하고, 제어용 소프트웨어 시스템이 탑재됨에 따라 소프트웨어의 안전성에 대한 중요도가 높아지고 있다. 다양한 산업 분야에서 소프트웨어가 사용되면서 소프트웨어에 의한 사고의 위협도 높아지기 때문에 소프트웨어 오동작에 의한 안전성 위협이 큰 이슈로 떠오르게 되었다. 소프트웨어의 사고는 사용자의 오조작에 의해서 발생할 수 도 있지만 가장 근본적으로는 설계 과정에서의 안전성에 대한 검증이 제대로 이루어지지 않아서 발생하게 된다. 따라서 본 논문에서는 소프트웨어 개발 프로세스에서 소프트웨어 안전성 분석 및 관리 활동이 어떻게 이루어져야 하는 가를 제시한다. 특별히 프로토타입이나 점진적 개발 프로세스에서의 안전성 분석 및 관리 활동의 적용 방안에 대하여 제시한다.

키워드 : 소프트웨어 안전성, 소프트웨어 개발 프로세스, 안전성 분석, 안전성 관리

Abstract As most devices in a wide range of automotive, aerospace, and missile have built-in software that controls the system behaviors, the safety of the software is growing in its importance. That is, the software safety has emerged as one of big issues because the threat of accidents caused by software malfunction is rising. Accident by software can be occurred from user mal-operation, but the fundamental reason of the accident comes from insufficient verification of the safety in software development process. Therefore, this paper presents how the software safety analysis and management activities should be done in the development process. In particular, we propose how to apply the safety analysis and management activities in the prototype or incremental development process.

Key Words : Software Safety, Development Process, Safety Analysis, Safety Management

1. 서론

현대 사회에서는 자동차, 철도, 항공, 원자력, 국방 등의 다양한 분야에서 소프트웨어가 접목되어 사용되고 있다. 이러한 소프트웨어들은 우리 생활에 편의를 위한 기능을 제공하지만, 생활에 밀접하게 관련이 있

Received 2016-02-12 Revised 2016-03-03 Accepted 2016-03-04 Published 2016-03-31
본 연구는 기초연구지원사업(NRF-2014R1A1A4A01005566)과 차세대정보통신개발사업(NRF-2015M3C4A7030505)의 지원에 의해 수행되었음.
*Corresponding author : Jang-Eui Hong (jehong@chungbuk.ac.kr)

는 만큼 소프트웨어의 오류는 크고 작은 사고를 야기하는 원인이 될 수 있다. 자동차 분야에서는 2014년 도요타 프리우스 자동차 리콜 사례가 있었다[1]. 철도 분야에서는 KTX의 사고사례를 들 수 있고, 항공우주 분야에서는 Arian 5 Rocket 폭발과 2012년 러시아 화성 탐사선의 추락사고가 있었다[2-4]. 원자력 분야에서 가장 대표적으로 알려진 Therac -25 사고가 있었다. 이러한 사고들은 모두 소프트웨어의 결함으로 비롯되었다[5].

소프트웨어 중심사회가 되어가면서 대부분의 국가기반시설 및 대단위 산업분야에서 소프트웨어를 이용한 제어가 이루어지고, 금융, 자동차, 항공, 전력, 국방, 의료, 교육 등 대부분 분야에서 소프트웨어 의존도가 높아짐에 따라 이의 오류로 인한 사고의 피해 범위와 규모가 확대되고 있다. 소프트웨어에 의한 사고를 예방하기 위한 소프트웨어 안전성에 대한 연구는 소프트웨어 개발 단계에서 매우 중요한 부분이다.

안전성이란 사고나 손실로부터 자유로운 상태라고 정의할 수 있는데, 소프트웨어 시스템의 경우 외부의 잘못된 입력으로부터 시스템을 안전하게 지켜주고, 재난 발생을 막기 위한 통제를 수행할 수 있는 가에 대한 관심을 말한다. 이와 관련하여 신뢰성과 안전성은 소프트웨어에 있어서는 매우 유사한 개념으로 사용되었으나 최근에는 그 개념이 분리되는 추세에 있다. 신뢰성은 특정한 환경 조건에서 준비된 기능이 특정 시간 내에 바르게 수행될 수 있는지에 대한 가능성을 말하는 것이라면, 안전성은 외부에서의 잘못된 입력으로부터 재난이 일어나지 않도록 할 수 있는 조건과, 계획된 기능이 수행될 것인지 아닌지에 대한 가능성을 의미 한다[6,7].

소프트웨어에 의한 사고는 조작 미숙 혹은 외적인 요인에 의해서 발생할 수도 있지만, 가장 근본적인 원인은 소프트웨어 개발단계에서의 안전성에 대한 검증이 제대로 이루어지지 않기 때문에 발생하게 된다. 소프트웨어 분석 및 설계 단계에서 안전성에 대한 연구는 대부분 안전성 분석 및 평가, 또는 소프트웨어 아키텍처 설계기법을 통한 안전성의 확보를 목표로 수행하는 것이 일반적인 추세이다. 가장 많이 알려진 소프트웨어 안전성 분석 기법인 고장유형 및 영향분석(Failure-Mode & Effect Analysis, FMEA)와 결함 위험분석 (Fault Hazard Analysis, FHA), 결함 트리분석 (Fault Tree Analysis, FTA)[8,9,10], 위험 및 운용 분석 (Hazard and Operability analysis, HAZOP) 등이 있다[8-14]. 이러한 다양한 연구들이 있지만, 소프트웨어 개발 프로세스에서 이러한 다양한 기법들을 어떻게 활용할 것인지에 대한 가이드라인은 충분하지 않다. ISO 26262 등과 같은 안전성 관련 표준들이 소프트웨어 개발에 대한 가이드라인을 제공하고 있지만, 구체적이지 못하며 대체적으로 폭포수 모델(Waterfall Model)과 같은 프로세스에서의 적용을 기준으로 제시하고 있다[11,12]. 따라서 본 연구에서는 먼저 소프트웨어 개발 프로세스에서 안전성을 어떻게 관리해야 하는가를 살펴보고, 이를 기반으로 점진적 개발에서의 안전성 분석 및 관리 방안을 제시한다.

본 논문의 구성은 다음과 같다. 2장에서는 배경 및 관련 연구로써, 소프트웨어 안전성과 관련된 사고 사례, 소프트웨어 안전성 표준, 그리고 안전성 관련 기존의 연구들을 살펴본다. 3장에서는 소프트웨어 시스템의 안전성 분석을 위한 다양한 기법을 정리하고, 4장에서는 소프트웨어 시스템 개발 단계별 안전성 관리 활동에 대하여 정리하였다. 5장에서는 점진적 소프트웨어 개발 절차상에서의 안전성 분석 및 관리 활동의 적용 방안을 제시하고, 마지막으로 6장에서 결론 및 향후 연구 내용을 제시한다.

2. 배경 및 관련연구

2.1 안전성 관련 사고사례

소프트웨어가 다양한 분야에서 활용되고 있는 만큼 여러 분야에서의 소프트웨어 안전성 관련 사고사례가 있다. 항공우주분야의 대표적인 사례는 Arian 5 Rocket의 폭발사고와 최근에 있었던 러시아 화성 위성 탐사선 '포보스-그룬트'호의 추락사고가 있다[3,4]. Arian 5 로켓 사고는 Arian 5호가 Ada로 작성된 Arian 2의 일부 모듈을 그대로 재사용 하며 발생하게 되었다. 문제가 된 모듈은 16비트 정수 값을 처리할 때는 문제가 없으나, 64비트 부동소수 값을 처리할 때 오버플로우(Overflow)가 발생하였고, 이를 제대로 처리하지 못해 사고가 발생한 것이다. 포보스-그룬트 호의 추락은 프로그램상의 문제로 인하여 탐사선 컴퓨터 2개 채널이 동시에 재부팅되면서 발생하게 되었다.

자동차 분야의 사례로는 최근 도요타 프리우스의 리콜 사고를 들 수 있는데, 리콜 원인은 전력 제어장치 소프트웨어의 결함으로 주행 중 차량이 갑자기 멈출 가능성이 발견되었기 때문이다[1]. 원자력 분야에서는 대표적으로 Therac-25의 사고 사례를 들 수 있는데, Therac-25는 암 환자용 방사선 치료기로서, 높은 에너지 방사광선을 빠르게 집중적으로 조사해 악성 종양을 파괴하는 장비이다[5]. 간호사와 방사선사가 환자에게 맞는 치료방식을 설정하기 위해 기계를 조작하는 과정에서 조작자가 입력 실수를 유발했고, 이것이 인터페이스에 저장되었는데 재입력이 8초 이내에 이루어지는 바람에 기계가 오 작동되었다. 6명의 사망자가 발생하였으며, 이 사고는 인터페이스 설계 부실과, 기계 조작에 대한 방사선사의 교육이 제대로 이루어지지 않은 데 원인이 있었다. 국방 분야에서는 패트리어트 미사일 사고가 있는데 이 사고는 소프트웨어 버그에 의해 사우디아라비아의 더란에서 28명의 미국 병사들이 사망했던 사건이다[13]. 마지막으로 철도분야의 사고로는 KTX의 정지사고 등이 있는데 KTX의 사고는 소프트웨어에 의한 사고비율이 60%에 이를 정도로 소프트웨어 사고가 큰 비중을 차지하고 있다.

2.2 관련 표준

다양한 산업분야에서 소프트웨어 안전성과 관련된 표준들이 제정되어 있다. 각 산업분야별 표준은 Table 1과 같고, Fig. 1은 안전성 표준들이 다루고 있는 주요 내용을 자동차 분야의 기능 안전성 표준인 ISO 26262를 기준으로 나타낸 것이다[11].

안전성 관련 표준들을 산업분야별로 살펴보면 항공분야의 안전성 평가에 대한 가이드라인이 가장 먼저 제정되었고, 국방 분야, 철도 분야, 원자력 분야가 안전성에 대한 관심이 먼저 이루어진 것을 알 수 있다. 최근에는 일반 사용자를 중심으로 하는 자동차 및 의료분야의 안전성이 중요시 되고 있는 실정이다.

이중에서 전기 및 전자장치의 가장 기본적인 표준이며, 공통 표준인 IEC 61508은 산업에 적용되는 안전성 규칙의 국제표준이다[14]. 이것의 이름은 전기/전자/프로그램 가능한 전자 안전 관리 시스템의 기능 안전으로 명명되었다. IEC 61508에서는 안전 관련 시스템의 정확한 기능 수행, 다른 안전 시스템과 외부 위험 감소 장치에 의존하는 제어 장비, 그리고 제어 장비를 제어하는 시스템의 부분적 또는 전반적인 안전으로 기능 안전을 정의하고 있다.

ISO 26262 또는 자동차 기능 안전성 국제 표준은 자동차에 탑재되는 E/E (Electric & Electronic) 시스템의 오류로 인한 사고방지를 위해 ISO에서 제정한 자동차 기능 안전 국제 규격이다[11]. ISO 26262는 프로세스 모델과 함께 요구되는 활동, 유무형의 증거물, 그

Table 1. International Standards for Safety by Industrial Areas

Domains	Standards	Titles of Standards (Publication Year)
Common	IEC 61508	Standard for Functional Safety of Electrical/Electronic/ Programmable Electronic Safety-related Systems (E/E/PE, or E/E/PES) (2010)
Automotive	ISO 26262	Road vehicles - Functional safety (2011)
Railway	IEC 62278	Railway applications - Specification and demonstration of reliability, maintainability and safety (RAMS) (2002)
	IEC 62279	Railway applications - Communications, signalling and processing systems - Software for railway control and protection systems (2002)
Airborne	ARP 4761	Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment (1996)
	DO-178C	Software Considerations in Airborne Systems and Equipment Certification (2012)
Nuclear Power	IEC 60880	Nuclear power plants - Instrumentation and control systems important to safety - Software aspects for computer-based systems performing category A functions (2006)
	IEC 62138	Nuclear power plants - Instrumentation and control important for safety - Software aspects for computer-based systems performing category B or C functions (2004)
	IEEE7-4.3.2	IEEE Standard Criteria for Digital Computers in Safety Systems of Nuclear Power Generating Stations (2010)
	IEC 61513	Nuclear power plants - Instrumentation and control important to safety - General requirements for systems (2011)
Defense	MIL-STD-498	Software Development And Documentation (1994)
	MIL-STD-882E	Department of Defense Standard Practice for System Safety (2012)
Medical	IEC 60601	Medical electrical equipment (2014)
	IEC 62304	Medical device software - Software life cyce processes (2006)
	ISO 13606	Health informatics - Electronic health record communication (2008)

리고 개발과 생산에 사용되는 안전성 지원 방법을 정의한다. 기능안전은 각 제품 개발 단계에 통합되어 고려되는데, 그 범위는 명세부터 시작하여 설계, 구현, 통합, 검증, 인증, 그리고 생산 및 인도 단계에 이른다. ISO 26262 표준은 기능 안전 표준 IEC 61508을 자동차 전기/전자 시스템에 적용시킨 것이다. ISO 26262는 모든 자동차용 전기/전자 안전 관련 시스템의 제품 수명 전 주기에 걸쳐 적용 가능한 자동차용 장비의 기능 안전을 정의한다.

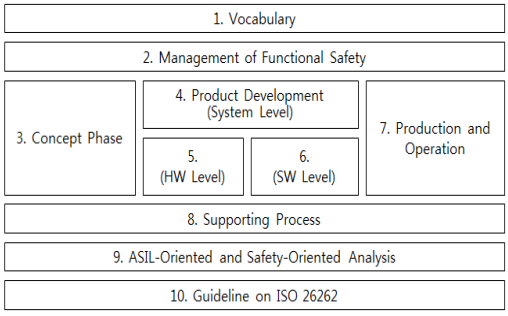

Fig. 1. Main contexts of ISO 26262

2.3 관련연구

소프트웨어의 안전성에 관한 연구는 다양하게 이루어지고 있다. Kim이 수행한 안전성 평가에 관한 연구에서는 안전 필수 시스템에서 소프트웨어 안전성을 보다 엄격하게 평가할 수 있는 방안을 제시하였다[15]. 상호보완적인 관계에 있는 결함분석 기법을 이용하여 안전성을 분석하는 방법으로 FTA와 FMEA 분석 기법을 결합한 하이브리드 방법을 제안하였다.

Hwang의 연구에서는 기존의 자동화된 소프트웨어 테스트 도구를 확장하는 방식을 통한 안전성 점검방안을 제시하였으며, 소프트웨어 개발 주기에서 파생된 안전성 활동의 결과들을 입력으로 표준에서 요구하는 평가항목들을 동적으로 테스트하는 평가도구를 제안하였다[16]. 소프트웨어 설계단계에서 수행한 안전성 활동 결과를 테스트 도구의 입력으로 사용하여 지속적으로 안전성을 검증하도록 기능을 확장하였다.

Park의 연구에서는 안전 필수 소프트웨어개발 시 구현 단계 또는 유지보수 단계에서 적용할 수 있는 Introduce De-Bouncing 기법을 제안했다[17]. 해당 기법은 스위치 바운싱 현상을 소프트웨어를 통해 해결할 수 있고 더불어 저렴한 비용으로 기존에 완성된 안전 필수 소프트웨어의 안전성을 향상 시킬 수 있는 기법이다. Kwon의 연구에서는 리소스 맵(Resource map)과 Fault Prevention Tree를 사용하여 사고 예방 동작간의 충돌 가능성을 분석하고, 이를 해결하기 위한 방안을 제안하였다[18]. 이는 Safety Critical 시스템에서 피해를 예방하기 위한 서브시스템들을 더욱 견고하게 설계할 수 있고, 시스템의 안전성이 향상될 수 있는 방법이다.

3. Software Safety 분석 기법

3.1 모델 검증 기법 (Model Checking)

모델 검증 기법은 정형기법의 한 종류로, 정형 언어로 작성된 모델이 검증하고자 하는 특성을 만족하는지 여부를 입증하는 방법이다[19]. 모델 검증 기법은 모델 명세와 입증할 검증 특성을 입력받아 모델 명세가 기능적 특성 및 안전성 요구사항을 만족하는지 논리적으로 증명한다. 만약 입력받은 모델이 검증 특성을 만족하지 못한다면 어떠한 상황에서 모델이 해당 검증 특성을 만족하지 못하는지 반례를 출력한다. 시스템 개발자는 이를 토대로 모델을 분석하고 보완할 수 있다.

3.2 결함 트리 분석(FTA)

안전 필수 시스템을 위한 FTA는 위험요소를 확인하는 것이 아니라 위험요소들의 원인을 분석하기 위한 수단이다[18]. FTA는 위험이 많은 사건을 구성할 수 있는 개별적인 오류들의 결합을 설명하기 위하여 부울 논리를 사용한다. 트리의 각 단계는 그 단계의 위쪽에 나타난 문제의 원인이 될 수 있는 필요 충분한 사건들을 식별하여 트리를 구성하는 과정으로 결함을 목록화한다.

3.3 결함 위험 분석(FHA)

결함 위험 분석은 위험에 대한 정성적 분석으로만 사용하거나, 필요시 정량적인 분석 방법으로 확장시킬 수 있는 연역적 방법이다[20]. FHA는 서브시스템이 갖는 세부 위험 경향, 위험의 원인, 그리고 서브시스템의 운영에 대한 위험의 영향을 결정하기 위해 사용한다. FHA는 반드시 치명적인 모드의(Catastrophic Mode) 위험과 허용 범위 밖의 (Out-of-Tolerance Mode) 위험

에 대한 양쪽 모두의 고려가 필요하다.

3.4 고장유형 및 영향 분석(FMEA)

고장유형 및 영향 분석 기법은 신뢰성/안전성 공학자들이 장비의 신뢰성을 예측하기 위해서 개발되어졌으며, 부품 그 자체에 고장 발생 원인이 삽입되는 것을 피하기 위한 목적으로 사용하는 기법이다[21]. FMEA는 설계, 공정, 품질보증 등 각 부분에 산재한 안전성 문제점을 정량적으로 관리하기 위한 기법이며, 점차 복잡해지는 안전성 관련 문제를 제품 개발 초기 단계에서 사전 제거하기 위한 목적으로 활용된다. FMEA의 단계는 (1) 모든 컴포넌트를 리스트 형태로 정의를 하고, (2) 각각의 고장 모드가 영향을 미칠 모든 컴포넌트, 시스템을 정의하고, (3) 각각의 고장 모드의 가능성과 함께 심각성을 계산하는 것이다[22].

FMEA의 결과물은 하나의 시스템 기능 별로 각 요소가 발생시킬 수 있는 잠재적 실패들과 그 실패로 인해 생기는 영향들을 도표로 기록하고 있으며, 영향도, 확률, 발생빈도, 위험도 우선순위 등으로 평가를 한 결과물들이 첨가되어 있다.

3.5 위험 및 운용 분석 (HAZOP)

HAZOP은 설계 또는 운용상에서 의도한 기능에서 벗어나 사고가 발생하는 것을 가정하고, 설계부터 예상 운용에서 일어날 수 있는 모든 가능한 일탈 상황과 그와 관련된 위험 요소를 찾으려고 하는 것이다[22,23]. 분석 과정에서 시스템에 대한 효율적인 검토를 수행하기 위하여 시스템 설계의 일정 구간을 분할하여 단계적으로 장비의 오작동이나 운전 조작 실수 등과 같은 위험 및 운영성을 평가한다.

3.6 요약

소프트웨어 개발에 사용될 수 있는 안전성 분석 기법은 지속적으로 개발되고, 개선되고 있기 때문에 그 종류와 수가 다양하다. 위에서 설명한 다섯 가지 기법은 소프트웨어 개발에서의 안전성 검증에 공통적으로 많이 사용되는 기법들을 대표적으로 나타낸 것이다. Table 2에서는 이외에도 그 동안 연구되어 왔던 다양한 안전성 분석 기법들이 위험 분석을 어느 정도 지원하는가, 소프트웨어 개발의 어느 단계에 적용할 수 있는가, 위험 분석의 접근 방법은 연역적인지, 귀납적인지 등에 따라 정리한 것이다. 정리된 것과 같이 다양한 안전성 분석 기법이 존재하며, 각 개발 단계에 맞는 합당한 분석 기법을 선택하여 적용하는 것이 중요하다.

Table 2. Applying strategies of safety analysis techniques

Technique Name	Hazard Analysis	Applying Phase[1)]	Results[2)]	Methods
Cause consequence analysis	Full supported	PD~DD	Q1/Q2	Deductive
Common cause failure analysis	Partially supported	PD~DD	Q1	Deductive
Event tree analysis	Partially supported	PD~DD	Q1/Q2	Deductive
Failure mode and effects analysis	Partially supported	PD~DD	Q1	Inductive
Fault hazard analysis	Full supported	PD~DD	Q1	Inductive
Fault tree analysis	Partially supported	PD~DD	Q1/Q2	Deductive
Functional failure mode & effect analysis	Partially supported	CD	Q1	Deductive
Hazard and operability analysis	Full supported	PD~DD	Q1	Inductive
Markov analysis	Partially supported	PD~DD	Q1/Q2	Deductive
Petri net analysis	Partially supported	PD~DD	Q1/Q2	Deductive
Preliminary hazard analysis	Full supported	CD~PD	Q1	Induc./Deduc.
Preliminary hazard list analysis	Full supported	CD~PD	Q1	Inductive
Safety requirement/criteria analysis	Partially supported	PD~DD	Q1/Q2	N/A
Sneak circuit analysis	Partially supported	DD	Q1	Deductive
Software safety assessment	Full supported	CD~PD	Q1	N/A
Subsystem hazard analysis	Full supported	DD	Q1	Induc./Deduc.
System hazard analysis	Full supported	PD~DD~T	Q1	Induc./Deduc.
Systems theoretic process analysis	Not Supported	CD~PD	Q1	Deductive

Remark 1) CD: Conceptual Design, PD: Preliminary Design, DD: Detailed Design, T : Testing
Remark 2) Q1: Qualitative, Q2: Quantitative, N/A: Not Applicable

4. 소프트웨어 시스템 개발 단계별 안전성 관리 활동

ISO 26262를 기준으로 살펴볼 때, 안전성을 고려하는 소프트웨어 시스템의 개발 단계는 Fig. 2에서 나타난 것과 같이 개념 정립 단계(Concept Phase), 제품개발 단계 (Product Development Phase), 그리고 운영유지 단계(Operation and Maintenance Phase)로 구분한다. 개념 정립 단계에서는 안전성에 대한 요구사항 및 이의 분석 활동이, 그리고 개발 단계에서는 안전성을 고려하는 하드웨어 및 소프트웨어에 대한 개발 활동이, 마지막으로 운영 단계에서는 안전한 소프트웨어 시스템의 운영을 모니터링하고 그 결과에 따른 피드백이 이루어진다. 각 단계별로 이루어지는 안전성 분석 및 관리 프로세스는 다음과 같다.

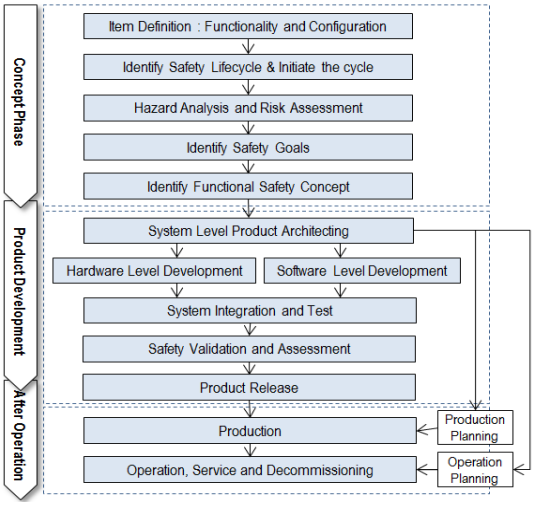

Fig. 2. Analysis and management Process for software safety

4.1 개념화 단계

개념 정립 단계에서는 안전성을 요구하는 시스템에 대한 정의 및 기능 요구사항에 대한 형상 정의를 시작으로, 요구사항을 기반으로 하는 위협 분석 및 안전성 목표(Safety Goals)를 식별하는 활동 등으로 이루어진다.

위협 분석은 운영 환경 조건, 시스템과 사용자의 상호작용 등을 고려하는 모든 가능한 시나리오에 대하여 분석되어야 하며, 위험한 상황이 발생했을 때, 이러한 상황이 사용자 또는 정의된 시스템에서 제어 가능한 상태인지, 얼마나 심각한 위험인지, 얼마나 빈번하게 발생하는지 등을 고려하여 분석한다. 위협 분석의 결과를 기반으로 시스템이 제공해야 하는 기본적인 안전성 요구사항을 확정하게 된다.

안전성 요구사항은 발생 가능한 모든 위험으로부터 사람의 생명을 보호하고 인재 물적 피해를 최소화 하는 방향으로 설정 된다. 설정된 요구사항은 기능 안전 요구사항으로 확정하여 제품 개발 단계로 전달된다.

4.2 제품 개발 단계

제품 개발 단계에서의 활동은 크게 5가지의 기본 활동으로 구성 된다; (1) 시스템 수준에서의 제품 설계 및 아키텍처 설계, (2) 하드웨어 수준의 제품 개발, (3) 소프트웨어 수준의 제품 개발, (4) 개발된 하드웨어와 소프트웨어의 통합 및 테스트, 그리고 (5) 기능 안전성에 대한 검증 및 평가 활동이다.

시스템 수준에서의 설계 활동은 상위 수준에서 시스템의 형상을 기반으로 시스템 아키텍처를 개발하는 활동이다. 이때 시스템 아키텍처는 선행적으로 명세된 기술적 안전성 요구사항을 고려하여 개발되어야 한다. 시스템 수준에서 정의된 기술적 안전 요구사항은 하드웨어 및 소프트웨어 개발 활동으로 전달된다. 하드웨어 수준의 제품 개발 활동은 전달된 안전 요구사항을 하드웨어 측면에서 명세하고, 이를 기준으로 제품 개발 활동을 진행한다. 이러한 활동은 일반적인 하드웨어 개발 절차를 준수하되, 안전성 목표를 준수하고 있는지를 점검하고 확인하는 활동을 반복적으로 수행한다.

소프트웨어 수준의 제품 개발 활동은 먼저 시스템 수준에서의 안전성 요구사항을 기반으로 소프트웨어 안전 요구사항을 명세한다. 즉 시스템 수준의 요구사항을 소프트웨어 안전 요구사항으로 구체화하면서 소프트웨어 개발 과정에서 고려되어야 하는 요구사항으로 분리해 내는 활동을 수행한다. 이러한 요구사항이 정의되면, 전체적인 소프트웨어 시스템에 대한 아키텍처 설계가 진행된다. 이러한 아키텍처는 안전성을 고려하는 소프트웨어 컴포넌트가 반드시 포함되도록 개발되어야 한다.

아키텍처를 이루는 각각의 소프트웨어 모듈은 상세 설계 과정을 거쳐 구현되며, 단위 테스트 과정을 거쳐 통합된다. 그리고 마지막으로 통합된 소프트웨어 시스템에 대하여 안전 요구사항이 만족되는지를 점검한다. Fig. 3은 시스템 수준의 요구사항에서 소프트웨어 요

구사항의 검증에 대한 개념적 흐름을 나타낸다.

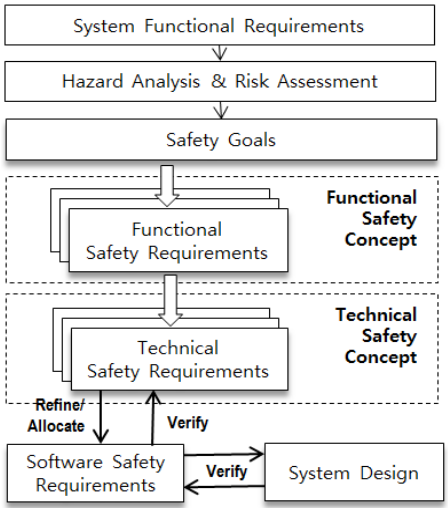

Fig. 3. Application of SW safety requirements

4.3 제품 운영 단계

제품이 생산되어 사용자에게 이관되면, 소프트웨어가 탑재된 시스템이 실제 환경에서 운영되게 된다. 그러나 이러한 안전성 분석 및 관리 활동들이 개념화 단계 및 개발 단계에서 심도 있게 적용되었다고 하더라도, 앞서 2장에서 살펴본 것과 같이 실제 운영 환경에서 많은 사고가 발생할 수 있다. 따라서 운영 단계에서도 시스템의 동작 과정에서 기능 안전 요구사항이 충분히 만족되는지를 모니터링 해야 한다. 모니터링 결과들은 시스템 내부에 존재하는 로그 형태로 기록되어야 하며, 예상하지 못했던 운영 조건에서 사고가 발생하면, 로그를 기반으로 어떠한 기능적 오동작이 있는지를 분석하는 과정을 수행한다.

이러한 분석 과정을 거쳐 Fig. 2에 나타난 안전성 관리 수명주기의 해당 단계로 이동하여 새로운 안전성 요구사항을 포함하는 개선 활동이 이루어진다.

5. 점진적 소프트웨어 프로세스에서의 안전성 관리 활동

소프트웨어 개발은 다양한 프로세스에 의해 개발될 수 있다. 기본적인 폭포수 모델을 근간으로 하여 반복적 개발 방식(Iterative Approach)과 점진적 개발 방식(Incremental Approach)이 안전 소프트웨어 개발에 적용될 수 있다. 반복적 개발과 점진적 개발의 개념적 차이는 Fig. 4에서 나타난 것과 같이 시스템의 개발하는 대상 모듈을 분리하여 개발하는가 아니면 추상화 수준에 따라 개발을 진행할 것인가에 따라 다르게 나타난다.

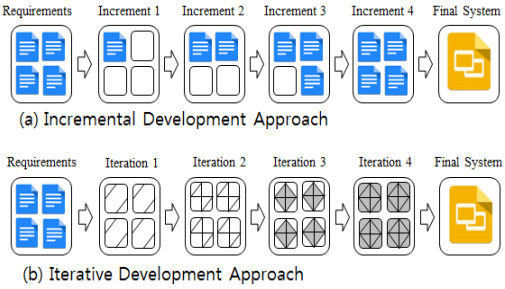

Fig. 4. Incremental and Iterative development approach

Fig. 4에서와 같이 점진적 개발 방식에 의한 안전 소프트웨어 개발에서는 도출된 소프트웨어 안전성 요구사항이 각 분할된 노듈에 할당되어 개발된다. 다만 각 모듈이 개발될 때 반드시 고려되어야 하는 사항은 각 모듈이 갖는 기능적 요구사항이 이외에 두 개의 모듈 간에 존재할 수 있는 상호작용에 대한 모듈이 임의로 고려되어야 한다는 것이다. 즉 Fig. 5에서와 같이 두 개의 모듈 사이에 존재하는 안전성 관련 요구사항이 정의되고 검증되어야 한다.

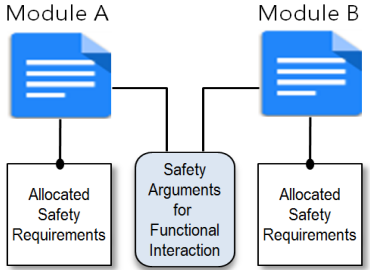

Fig. 5. Safety consideration in incremental approach

반복적 개발 접근 방법에서는 시스템에 부여되는 안전성 요구사항들이 계층적 구조를 가지고 설계되어야 한다. 일반적인 결함 트리 분석에서처럼 상위 수준에서의 결함이 하위수준에서의 결함으로 구성될 수 있다는 기본 개념을 적용하는 것이다. 하위 수준에서의 결함이 상위 수준의 결함을 유발하기 위해 AND 조건

으로 연결될 것인가 OR 조건으로 연결될 것인가를 고려하여 안전성 요구사항을 할당하고 검증한다.

이와 같은 접근 방법에 의해 안전 소프트에어의 개발 프로세스가 점진적 방식을 적용하는가, 아니면 점진적 개발 방식을 적용하는 가를 적용하는가에 따라 안전 요구사항의 할당 및 적용에 다소 차이가 있을 수 있다. 기본적으로 소프트웨어 개발 과정에서 사용될 수 있는 나선형 모델이나, 애자일(Agile) 방법에서도 제시한 접근 방법을 통해 안전 요구사항을 충족시키는 소프트웨어 개발이 가능할 수 있다.

6. 결론 및 향후 연구내용

소프트웨어 안전성에 대한 분석은 안전 필수 시스템의 개발에서 반드시 수행되어야 하는 활동 중 하나이다. 산업분야별 별도의 안전성 관련 표준이 정의되어 있고, 이에 따른 필수 안전 소프트웨어 개발이 가이드 되고 있지만, 대부분의 시스템은 소프트웨어의 동작에 따라 작동되기 때문에 시스템에 내장되는 소프트웨어의 동작 및 제어를 어떻게 안전하게 수행해야 할 것인가가 안전 필수 시스템 개발에서의 가장 핵심적인 요소라고 볼 수 있다. 따라서 소프트웨어의 개념화 단계부터 개발 단계 및 운영 단계에서 안전성은 명확하고 일관성 있게 분석 및 관리 되어야 한다.

본 논문에서는 소프트웨어 개발 프로세스 단계에서 어떠한 안전성 관련 활동 등이 있는가를 정리하였고, 이를 통해 안전 소프트웨어 개발 활동이 어떻게 이루어지는 가를 프로세스 관점에서 조명하였다. 이를 위해서는 먼저 정확한 기능 요구사항으로부터 안전성 목표를 정의하고, 이에 따른 안전성 요구사항을 도출하는 것이 중요하다. 이러한 안전 요구사항들이 시스템의 설계 활동에서 반영될 수 있도록 고려되어야 함은 물론, 시스템의 개발 과정에서 반드시 기능 안전성에 대한 검증이 수행되어야 한다. 또한 본 논문에서는 점진적인 개발 프로세스를 기반으로 안전 필수 소프트웨어가 개발되는 경우, 어떠한 프로세스 관점에서의 고려가 있는지도 제시하였다. 이들은 안전 소프트웨어를 개발하는 조직에서 개발 프로세스를 정립하기 위한 기반으로 활용될 수 있을 것이다. 향후의 연구로써는 3장에서 제시한 다양한 안전성 분석기법이 소프트웨어의 특성에 따라 어떻게 적용될 수 있는가를 분류하고, 이에 대한 적용방안을 고찰하는 것이다.

ACKNOWLEDGEMENTS

This research was supported by Next-Generation Information Computing Development Program(NRF-2014M3C4A7030505), and also partially supported by Basic Research Support Program(NRF-2014R1A1A4A01005566) through the NRF of Korea.

REFERENCES

[1] Korea YeonHap News, *Toyota Prius Recall - Software Faluts*, 2014.

[2] MK News, KTX Stop Again, http://news.mk.co.kr/, 2011. 2.

[3] J. L. LIONS, Ariane 5 Flight 501 Failure, Report by the Inquiry Board, http://www.esa.int/For_Media/Press_Releases/Ariane_501_-_Presentation_of_Inquiry_Board_report, 1996. 7.

[4] A. G. Stephenson, L. S. Lapiana, D. R. Mulville, P. J. Rutledge, F. H. Bauer, D. Folta, G. A. Dukeman, R. Sackheim, P. Norvig, *Mars Climate Orbiter Mishap Investigation Board Phase I Report*, 1999.

[5] Nancy Leveson, *Software: System Safety and Computers*, Addison-Wesley, 1995.

[6] K. S. Kim and H. C. Kim, “Comparative Study for Software Reliability Model Based on Finite and Infinite Failure Property using Rayleigh Distribution,” *Journal of Digital Convergence*, Vol. 12, No. 12, pp. 277-284, Dec. 2014.

[7] M. H. Kim and Man-Gon Park, “A Study on the Software Fault Modes and Effect Analysis for Software Safety Evaluation,” *Journal of Korea Multimedia Society*, Vol. 15, NO.1, pp. 115-130, Jan. 2012.

[8] C. H. Han, Y. S. Lee, J. Ahn and W. S. Jo, "A study on the Correlation Hazard Analysis for Signaling System Safety," *Journal of the Korean Society for Railway*, pp. 634-641, 2007.

[9] E. S. Kim, S. H. Yoon and J. Yoo, "A Survey on Safety Analysis Techniques for Safety-

Critical Systems," *Journal of IT Conver gence Society for SMB*, Vol. 2, No. 1, pp. 11-18, Jun. 2012.
[10] C. A. Ericson, "Fault tree Analysis-A History," *Proceedings of the 17th International System Safety conference*, pp. 1-9, 1999.
[11] ISO Standards, *ISO 26262 Road vehicles-Functional safety*, ISO, 2011.
[12] I. Sommerville, *Software Engineering 8th Edition*, Addison-Wesley, 2007.
[13] United States General Accounting Office, *GAO Report IMTEC-92-26*, 1992. 2.
[14] IEC Standards, *IEC 61508 - Standard for Functional Safety of Electrical/Electronic/ Programmable Electronic Safety-related Systems, International Electrotechnical Commission*, 2010.
[15] M. H. Kim and M. G. Park, "A Study on the Software Fault Modes and Effect Analysis for Software Safety Evaluation," *Journal of Korea Multimedia Society*, Vol. 15, No. 1, pp. 115-130, Jan. 2012.
[16] J. G. Hwang, H. J. Jo and H. S. Kim, "Design of Train Control Software Safety Evaluation Tool," *Journal of the Korean Society for Railway*, Vol. 11, No. 2, pp.139-144, Apr. 2008.
[17] J. J. Park and J. E. Hong, "An Approach to improve software safety by Code refactoring," *Proceedings on Korea Computer Congress*, pp.532-534, 2013.
[18] J. J. Kwon, D. W. Kim, J. J. Park and J. E. Hong, "Collision Analysis of Safety Devices to Prevent Hazards in Safety Critical Systems", *IEEE Software Security and Reliability*, pp.245-254, 2014.
[19] E. M. Clarke, O. Grumberg and D. Peled, *Model checking*, MIT press, 1999.
[20] P. J. Wilkinson, *Functional Hazard Analysis for Highly Integrated Aerospace System*, Certification of Ground/Air System Seminar, 1998.
[21] H. H. Kim and N. H. Lee, "The Case Study on Software FMEA for the Efficient Improvement of Functional Safety," *Transactions of The KSAE*, Vol. 2012, No. 11, pp.1303-1308, Nov. 2012.
[22] I. B. Pirie, "Software ? How do we know it is safe?", *IEEE Conference on ASME*, pp.122-129, 1999.
[23] K. B. Sung and M. G. Park, "Safety Activities on The Software Life-Cycle," *Journal of Korea Multimedia Society*, pp.432-437, 1998.

저 자 소 개

김 순 겸(Soon-Kyeom Kim) [학생회원]

- 2015년 8월 : 충북대학교 컴퓨터공학과 (전산학학사)
- 2015년 9월 ~ 현재 : 충북대학교 컴퓨터과학과 석사과정

<관심분야> : 모델기반 소프트웨어 공학, 소프트웨어 안전성, 소프트웨어 품질

홍 장 의(Jang-Eui Hong) [정회원]

- 2001년 2월 : KAIST 전산학과 (전산학박사)
- 2002년 10월 : 국방과학연구소 선임연구원
- 2004년 8월 : ㈜솔루션링크 기술연구소장
- 2004년 9월 ~ 현재 : 충북대학교 소프트웨어학과 교수

<관심분야> : 소프트웨어공학, IT 융합, 소프트웨어 품질, 소프트웨어 안전성, 저전력 소프트웨어 개발