Journal of Korea Multimedia Society Vol. 21, No. 9, September 2018(pp. 1099-1109)
https://doi.org/10.9717/kmms.2018.21.9.1099

자동차 안전성을 위한 소프트웨어 FMEA 가이드라인

최준열[†], 김용길[††], 조준형[†††], 최윤자[††††]

The Software FMEA Guideline for Vehicle Safety

Junyeol Choi[†], Yongkil Kim[††], Joonhyung Cho[†††], Yunja Choi[††††]

ABSTRACT

Most of the automotive electronic systems are equipped with control software. ISO 26262 standard has been published to prevent unreasonable risk due to E/E system malfunction. And many automotive companies apply ISO 26262 for safe series product. In ISO 26262 standard, the product quality improves through deductive and inductive safety analysis in all processes including system and software development phase. However, there are few studies on software safety analysis than systems. In the paper, we study the software FMEA(Failure Mode Effect Analysis) technique for product quality of vehicular embedded software. And we propose an effective guideline of software FMEA as EPB industrial practice.

Key words: Safety Analysis, ISO 26262, Embedded Software, Software FMEA

1. 서 론

차량용 소프트웨어의 대부분은 제어 소프트웨어이며, 제어 소프트웨어 기능의 장애는 시스템의 안전 목표를 위반하는 원인이 되어 심각한 손실 또는 재난을 일으키게 된다.

안전 목표는 시스템 개발 시 HARA(Hazard Analysis and Risk Assessment)를 통해 도출되며, 안전 목표를 달성하기 위해 시스템의 개념단계에서 시스템의 안전 상태 설계, 기능 안전 요구사항 구체화, 시스템 구성 요소에 ASIL(Automotive Safety Integrity Level)을 할당하게 된다. 이후 기능 안전 요구사항을 달성하기 위해 시스템의 기술 안전 요구사항이 도출되고, 기술 안전 요구사항은 하드웨어와 소프트웨어 같은 구성요소에 할당되어 체계적인 개발이 진행된다[1].

개발자는 시스템, 하드웨어 및 소프트웨어의 개발 단계별로 안전 분석을 수행한다. 안전 분석에는 귀납적 분석기법과 연역적 분석기법이 있다. 주로 귀납적 분석기법으로 FMEA, 연역적 분석기법으로 FTA를 수행한다[17]. FMEA, FTA는 많은 산업분야에서 사용되는 방법임에도 불구하고 소프트웨어 산업에서 사용하기에 다소 문제점이 존재한다. 특히 하위 컴포넌트 수준으로 분석되어야 하는 소프트웨어 FMEA에서 일반화된 FMEA 가이드라인을 따르게 되면 문제점이 발생한다. 이는 차량용 임베디드 소프트웨어의 특성을 이해하지 못하고, 시스템 개발단계에서 사용하던 FMEA 가이드라인에 따라 안전 분석함으로써 소프트웨어의 근본적인 위험요소를 잘못 식별한 것으로 보여 진다.

※ Corresponding Author : Junyeol Choi, Address: (13486) 21, MANDO Global R&D, Pangyo-ro 255beon-gil, Bundang-gu, Seongnam-si, Gyeonggi-do, Republic of Korea,, TEL : +82-10-6581-0123, FAX : +82-2-6188-3991, E-mail : trustcjy@gmail.com
Receipt date : May 2, 2018, Revision date : Jul. 29, 2018
Approval date : Jul. 29, 2018

[†] MANDO corporation, Global R&D, Brake Center
[††] MANDO corporation, Global R&D, Brake Center
(E-mail : yongkil.kim@halla.com)
[†††] MANDO corporation, Global R&D, Brake Center
(E-mail : joonhyung.cho@halla.com)
[††††] Dept. of Computer Eng, Kyungpook National University
(E-mail : yuchoi76@knu.ac.kr)

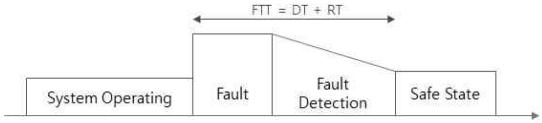


Fig. 1. System Operations.

본 연구와 연관되는 선행 된 프로젝트에서 차량용 임베디드 소프트웨어 FMEA가 비효율적임을 확인하였다. 비효율적이었던 원인은 불필요한 안전 메커니즘의 개수가 과다했던 것으로 분석되었다. 본 연구에서는 차량용 임베디드 소프트웨어의 특성에 적합한 소프트웨어 FMEA 가이드라인을 제안하는 것을 목표로 한다. 본 논문은 2장 안전 분석 및 소프트웨어 FMEA, 3장 문제점 분석, 4장 가이드라인의 제안, 5장 평가, 그리고 6장 결론으로 구성된다.

2. 안전 분석 및 소프트웨어 FMEA

IEC 61508, EN 50128, ISO 26262, IEC 60601 등의 기능 안전문서가 제정되면서, 소프트웨어의 안전 분석에 대한 중요성이 더욱 강조되고 있다. 안전 분석의 목적은, 시스템이 안전 목표를 지속적으로 만족시키거나 Fig. 1과 같이 안전 목표를 충족하지 못하는 경우에는 결함을 감지하고 위험이 발생하지 않도록 안전 상태(Safe state)로 전이되게 하는 것이다. 결함이 발생될 때, 시스템이 안전 상태로 전이되는 시간을 결함허용시간(Fault Tolerant Time)이라 한다. 결함허용시간은 감지 시간(Detection Time)과 반응시간(Reaction Time)의 합이다. 시스템은 소프트웨어와 하드웨어의 하위 컴포넌트들로 이루어져 있으며, 결함허용시간 내에 소프트웨어와 하드웨어는 시스템이 안전 상태로 전이되게 설계되어야 한다.

2.1 PHA(Preliminary Hazard Analysis)

PHA는 미 군용표준문서 MIL-STD-882에서 유래된 기법으로서 시스템 개발에 대한 정보가 적은 환경에서 안전 분석을 용이하게 하는 방법이다. 시스템 개념 설계단계에서 사용되며 위험성 분석(Fault Hazard Analysis), 제품 위험성 분석(System Hazard Analysis) 및 운용 위험성 분석(Operating Hazard Analysis) 등이 PHA 기법과 유사하다. PHA 분석기법은 시스템 개념설계단계에서 주로 사용되는 안전분석 기법이다[3].

2.2 FTA(Fault Tree Analysis)

FTA는 대륙 간 탄도 미사일 발사 제어 시스템을 평가하기 위해 Bell 연구소에서 고안되었으며 Boeing 社 등에서 최초 적용하였다. FTA는 연역적 분석의 대표기법 중 하나이다. 최상위 사상으로부터 해당 사상을 발생시키는 사건들을 논리게이트를 이용하여 하향식 형태로 표현하며 분석하는 기법이다. 체계적 결함분석으로 잠재적 원인을 도출할 수 있는 장점이 있으나, 분석에 많은 정보가 필요하고 결함 통제가 가능한 하위 시스템에 대한 분석은 불가능하다는 단점이 있다[4-7].

2.3 FMEA

FMEA는 NASA의 아폴로 프로젝트에서 처음으로 사용되었으며 미 군용표준문서 MIL-STD-1629 표준으로 제정되었다. 이후 자동차 산업에서 미국 대표 자동차 회사들에 의해 제정된 QS-9000이 일반산업에서 사용되었다. FMEA는 귀납적 분석의 대표 기법의 하나이다. 시스템을 구성하는 한 요소의 장애가 시스템 전체에 미치는 영향을 분석하여 장애를 피할 수 있도록 한다. 설계, 공정, 품질 등 각 분야에 내포된 문제를 개발단계 초기에서 제거하기 위한 목적으로 사용한다[8-10].

2.4 관련연구

현재 산업계에서는 소프트웨어 FMEA에 대한 중요성이 대두되고 있으며 일반적으로, 최소단위 컴포넌트까지 FMEA를 수행하도록 하고 있다. 안전 분석을 위해서 소프트웨어 설계 시점에 소프트웨어를 구성하는 컴포넌트 간 인터페이스가 분석되어야 한다. 인터페이스 분석이 완료되면 Fig. 2처럼 장애 유형을

Effects	S	Failure mode	Cause	Preventive action
EPB software error	8	Unauthorized applying operation(iPbcOutMotorCommandLeft==54)	Systematic fault by developer	Initial state: 2017-08-09
[Brake System] > Safe Goal_01 violation(ASIL				-[SM] Range check about output variables at CL component

Fig. 2. Software FMEA.

정의한다. 이후 장애 유형 발생의 원인과 영향성을 분석한 결과를 기준으로 RPN(Risk Priority Number) 평가 및 적절한 안전 메커니즘을 도출한다[10]. 도출된 안전 메커니즘은 소프트웨어 내부에 적절하게 적용되어야 하며 안전 메커니즘의 추가적인 적용은 개발시간과 소프트웨어의 복잡도를 증가시킨다. 자동차 또는 시스템 수준에서 안전 목표가 달성되는 최소한의 안전 메커니즘을 구현하는 것이 중요하다.

관련된 연구로 최근 자동차 산업의 기능 안전을 위해 ISO 26262 표준이 제정되었다[1]. 해당 표준에서는 소프트웨어 기능에 대한 문제를 FMEA 등의 안전 분석 기법을 사용하여 아키텍처 설계단계에서 보완하도록 요구한다.

또한 시스템의 대형화로 인한 복잡한 시스템은 문제점들을 사전에 식별하여 보완해야 할 필요성이 있다. 이러한 요구를 충족하기 위하여 모델 기반의 시스템 공학 방법론을 활용한 체계적인 분석 기법이 존재한다[15, 16]. M.H Kim은 안전 분석을 위해 단일 방법이 아닌 FTA 및 FMEA 기법을 결합하는 안전 필수 시스템 분석 방법을 제안하였다[14]. 검증 대상을 모델링하고 소프트웨어 FMEA를 자동화하여 분석하는 방법도 연구되었다[2].

위에서 언급된 FMEA 연구기법을 사용하면 소프트웨어 요구사항 또는 상위 컴포넌트에 대한 분석은 쉽다. 하지만 상향식 분석 방법이라는 특성 때문에 시스템의 구성요소 중 하나인 차량용 임베디드 소프트웨어, 그리고 소프트웨어를 구성하는 하위 컴포넌트에 대한 분석방법으로 사용하기에는 한계가 있다.

본 논문에서는 차량용 임베디드 소프트웨어 FMEA에 적합한 가이드라인을 제안하여 차량용 임베디드 소프트웨어 FMEA 사용상의 효율을 향상시키고자 한다. FMEA 종류 중 프로세스 측면의 FMEA를 제외한 기능측면의 DFMEA(Design FMEA)[10]를 대상으로 한다. 또한 소프트웨어 장애의 원인 중 동작 환경(ECU, RAM 등) 결함, 환경 오류(스케줄링, 지연 등) 등 랜덤 하드웨어 장애는 고려하지 않고 컴포넌트 개발자의 실수가 시스템 기능 장애로 이어지는 소프트웨어 설계 부분의 장애 유형만 다룬다.

3. 문제점 분석

ISO 26262에서는 소프트웨어의 결함을 모두 제거하는 것이 불가능에 가깝다고 한다. 따라서 소프트웨어 개발 시 시스템의 안전 목표를 위반하는 결함들이 예측되면, 해당 결함들이 자동차의 위험이 발생되지 않도록 소프트웨어를 설계해야 한다.

소프트웨어 설계영역에서 FMEA는 소프트웨어 아키텍처 설계가 완료되기 전에 수행되어야 하며 Fig. 3의 절차에 따라 반복적으로 수행된다. 기 수행된 프로젝트에서 도출되었던 안전 메커니즘을 동료검토 단계에서 재평가해보면 불필요한 안전 메커니즘이 과다했었다.

불필요한 안전 메커니즘이 과다했던 원인은 RPN 평가에 시스템에서 사용하던 FMEA 가이드라인을 따른 것과 하위 컴포넌트들을 동시에 안전 분석한 것으로 분석되었다. 해당 원인들로 소프트웨어가 체계적으로 분석되지 못했고 결국 과다한 안전 메커니즘이 만들어졌다.

RPN 평가에 사용되는 심각도 기준은 VDA volume 4[11], SAE J1739 또는 ISO 26262를 다루는 FMEA 문헌 자료 등을 참고하여 OEM과 협의되어 정의되거나, OEM과 협의가 어려운 경우 자료에 제시된 심각도 기준을 그대로 사용한다. 해당 자료에서 사용되는 심각도 기준은 Table 1[18]과 같이 자동차 환경의 최상위 시스템 기능에 대한 잠재적인 위험을 다루는 내용이다. 따라서 소프트웨어 하위 컴포넌트 개발자가 안전 분석 시 바로 사용하기에는 어려움이 있다.

소프트웨어 하위 컴포넌트 하나는 시스템 전체 구성요소 중 매우 작은 범위를 차지한다. 하위 컴포넌트 기능의 조합이 상위 컴포넌트의 기능이 되며, 상

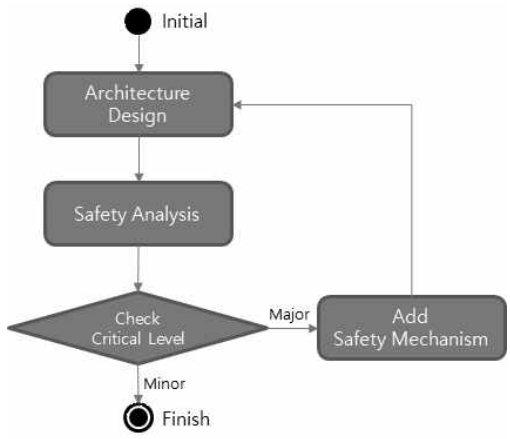


Fig. 3. The Process of Software FMEA.

Table 1. SAE J1739 FMEA Severity Table

Severity	SAE J1739 severity description
10	Very high severity ranking when a potential failure mode affects safe vehicle operation and/or involves noncompliance with government regulation without warning
9	Very high severity ranking when a potential failure mode affects safe vehicle operation and/or involves noncompliance with government regulation with warning
8	Vehicle/item inoperable, with loss of primary function
7	Vehicle/item operable, but at reduced level of performance. Customer dissatisfied
6	Vehicle/item operable. but Comfort/Convenience item inoperable. Customer experiences discomfort
5	Vehicle/item operable. but Comfort/Convenience item operable at reduced level of performance. Customer experiences some dissatisfaction
4	Fit & Finish/Squeak & Rattle item does not conform. Defect noticed by most customers.
3	Fit & Finish/Squeak & Rattle item does not conform. Defect noticed by average customer
2	Fit & Finish/Squeak & Rattle item does not conform. Defect noticed by discriminating customer
1	No Effect

위 컴포넌트 기능 조합으로 소프트웨어 요구사항이 충족된다. 그리고 소프트웨어와 하드웨어 기능의 조합으로 시스템 및 자동차 기능이 구현된다.

Fig. 4와 같이 시스템이 커질수록 소프트웨어의 구조는 복잡해진다. 복잡하고 거대한 소프트웨어는 프로젝트 기간을 고려하여 다수의 개발자가 나누어서 개발한다. 소프트웨어가 거대할수록 하위 컴포넌트 하나를 분석하는 개발자는 안전 분석을 진행하기 어려워진다. 하위 컴포넌트 장애 유형과 자동차 또는 시스템 동작의 연관성을 정확히 분석하는 것은 불가능에 가깝기 때문이다. 결국 소프트웨어 하위 컴포넌트를 시스템 FMEA 가이드라인에 따라 분석하면 컴포넌트 대부분은 과다한 안전 메커니즘을 가질 수밖에 없다.

자동차 산업에서 적절한 소프트웨어 FMEA가 어

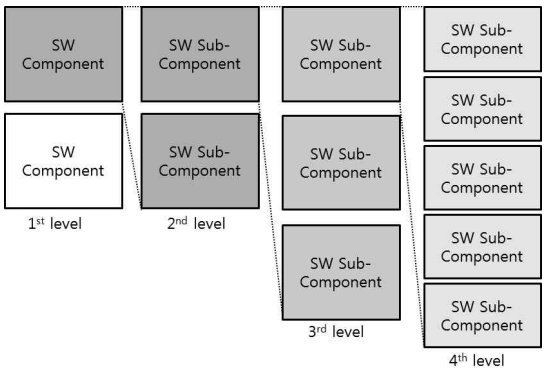


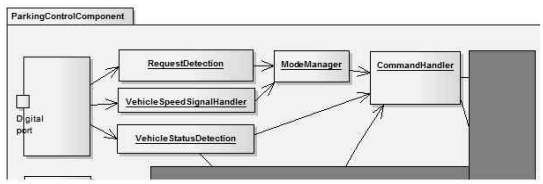
Fig. 4. The Level of Software Architecture.

렵다는 것을 전자식 주차브레이크 소프트웨어를 예로 들어 설명한다. 아래에 기술된 2가지 시스템 요구사항과 관련된 컴포넌트 및 기능은 Fig. 5와 같다.

요구사항 1. The feature shall be deactivated for vehicle speed greater than 15km/h.

요구사항 2. The clamp force shall be maintained during vehicle holding or parking.

Fig. 5에 언급된 5개 소프트웨어 컴포넌트 중 “Vehicle Status Detection” 컴포넌트를 제외한 4개 컴포넌트는 요구사항 1과 2를 만족하기 위한 구성요소이다. “Vehicle Status Detection” 컴포넌트는 시



RequestDetection	VehicleSpeedSignalHandler	VehicleStatusDetection
-Switch detection function	-WheelSpeed FL detection -WheelSpeed FR detection -WheelSpeed RL detection -WheelSpeed RR detection -VehicleSpeed estimation	-Vehicle Status confirmation -Engagement enable -Disengagement enable

ModeManager	CommandHandler
-Temperature estimation -ParkApply request -Dynamic request -Release request -Reclamp request	-Current status detection -Command confirmation -Left motor operation -Right motor operation -Left motor deactivation -Right motor deactivation -Warning indication

Fig. 5. Software Components and Functions.

스템 설계단계에서 요구사항 1을 위해 추가된 안전 메커니즘을 가지는 컴포넌트이다. 즉 “Vehicle Status Detection” 컴포넌트는 소프트웨어 안전 요구사항을 할당받은 컴포넌트이며, 타 컴포넌트들은 일반 기능 요구사항을 할당받았다.

Fig. 6은 “Vehicle Speed Signal Handler” 컴포넌트의 FMEA 결과 중 일부이다. 해당 컴포넌트의 기능 중 “Vehicle Speed Estimation” 기능은 요구사항 2를 만족하는 데 필요한 기능이다. 요구사항 1을 위해 15 km/h 판단에 사용될 필요는 없다. 이미 15 km/h 오판단에 대한 위험은 “Command Handler”에 설계되었기 때문이다.

“Command Handler” 컴포넌트의 기능은 Fig. 6에서 확인되듯이 “Vehicle Speed Signal Handler” 컴포넌트가 어떠한 잘못된 기능을 하더라도 “Vehicle Status Detection” 컴포넌트의 “Engagement enable” 기능 결과를 참고하여 요구사항 1의 위반을 막는 것이다. 하지만 “Vehicle Speed Signal Handler” 컴포넌트의 개발자는 장애 유형이 시스템 기능에 큰 영향을 준다고 잘못 분석하고 심각도 값을 높게 선정하였다. 잘못된 영향성 분석으로 “Input data trembling”이라는 원인 식별과 함께 의미 없는 “Data Saturation” 안전 메커니즘을 만들게 되었다.

하위 컴포넌트 개발자의 입장에서 장애 유형 하나와 자동차 또는 시스템 기능과의 영향성은 분석이 어려웠다. 시스템 개발자 역시 소프트웨어의 구조를 명확하게 이해하기 어려웠기 때문에 소프트웨어 FMEA의 적절성에 대한 검토는 쉽지 않았다. 소프트웨어는 과다한 안전 메커니즘을 가지게 되었으며, 그 결과 소프트웨어가 목표 실행시간을 초과하는 문제가 발생하였다. 안전 메커니즘의 적절성에 대한 평가 시 대부분의 안전 메커니즘은 시스템의 안전 요구사항과는 관계가 없었다.

부적절한 안전 메커니즘이 설계되는 문제점 외에도, 각각의 컴포넌트에 반영된 안전 메커니즘 간의 충돌로 시스템의 기능이 상실되는 문제가 있었다. 한 가지 예를 들면 후순위로 동작되는 컴포넌트에 기동작된 컴포넌트들의 결과를 비교하여 개연성을 확인(Plausibility check) 기능이 존재했다. 하지만 먼저 수행되는 컴포넌트의 개발자가 출력 신호에 대한 예외처리를 안전 메커니즘으로 적용하였고, 그 결과 비교 대상이었던 신호의 값은 후순위 컴포넌트 개발자의 예상을 벗어나게 되었다. 결국 후순위로 수행되는 컴포넌트에 적용되어있던 개연성 확인 기능은 정상적으로 동작하지 않았다.

4. 가이드라인의 제안

우리는 소프트웨어 설계 단계에서 개발자가 객관적이고 체계적인 안전 분석을 할 수 있도록 가이드라인을 제안하고자 하였다. 먼저 3장에서 언급한 문제점의 원인을 분석하였으며 Table 2와 같이 정리하였다.

문제의 원인을 해결하기 위해 소프트웨어 하위 컴포넌트 안전 분석 단계에서 시스템 또는 자동차의 영향은 생각하지 않도록 하였다. 해당 컴포넌트에 할당된 요구사항들만 고려하여 후순위 컴포넌트의 요구사항 충족 가능 여부만 분석되도록 하였다. 또한 분석 순서도 데이터 흐름을 기준으로 백-프런트 순서로 분석하도록 하여 후순위 컴포넌트에 대한 영향

Effects	S	Failure mode	Cause	Preventive action	O
System element: Vehicle Speed Signal Handler					
Function: Sending speed information					
[System element 1] Unintended Apply request at high speed	8	Incorrect wheel speed	[System element 3] Input data trembling	Initial state: 2015-12-23	
				Preventive action 1	10
				Revision state: 2016-02-11	
				-[SM] Data saturation by 15	8
			Systematic failure	Initial state: 2016-02-11	
				-[SM] Range check for In/Out data	6
System element: Command Handler					
Function: Sending actuation command					
[System element 1] Unintended Apply request at high speed	8	Sending actuation command	Wrong Actuation enableline from Vehicle Status Detection and Wrong apply request from Mode Manager	Initial state: 2016-03-08	
				-[SM] Vehicle Status Detection	2

Fig. 6. Existing Software FMEA.

Table 2. Causes for Issues

Issues	Cause
Excessive safety concept	Lack of understanding of the component developer for the entire software. Developers design excessive safety mechanisms as the lack of understanding.
Conflicts between safety mechanism	A simultaneous safety analysis of the components making up the software. Multiple components simultaneously analyze and derive safety mechanism. An unintended failure occurs due to a collision between safety mechanism.

성 분석이 정확하도록 하였다.

장애 유형이 위험하다고 평가되는 경우, 출력 신호를 전달받는 후순위 컴포넌트 개발자와 협의 후 안전 메커니즘을 추가하도록 하였다. 최종적으로 소프트웨어 FMEA의 효율성 및 객관성을 확보하기 위해 아래와 같이 지켜야 할 몇 가지 제약사항을 만들었다.

제약사항 1. 소프트웨어 FMEA는 소프트웨어 아키텍처에서 컴포넌트의 명확한 기능을 식별 및 관계를 분석 할 수 있는 배치 뷰[19] 또는 할당 뷰[20] 등에서 진행한다.

제약사항 2. 안전 분석 시 RPN값이 높더라도, 분석된 원인이 후순위 컴포넌트가 안전 요구사항을 위반하게 되는 유일한 원인이 아니라면 상위 단계의 설계자와 협의하여 안전 메커니즘을 반영한다.

제약사항 3. 컴포넌트의 안전 분석 순서는 외부로 송출되는 신호를 만들어 내는 컴포넌트부터 외부에서 입력되는 신호를 받는 컴포넌트까지 백 - 프런트 순서로 분석한다.

제약사항을 만들고 소프트웨어 FMEA을 위한 심각도 기준을 새롭게 정의하였다. 기존에 사용하던 심각도 자료는 차량에서 일어날 수 있는 위험성을 나타내고 있기 때문이다. 해당 심각도에 대한 정의는 자동차 또는 시스템 개발 단계에서 사용하기 적절한 내용이며 개발자 주관에 따라 심각도 값이 바뀔 수 있는 내용을 담고 있다.

먼저 소프트웨어를 개발하는 데 있어 가장 많이 발생하는 결함의 원인을 분석하였다. 크게 소프트웨어가 동작하는 하드웨어의 결함, 개발자의 실수 등이 있었다. 개발자의 실수를 발생시킬 수 있는 요소 중 큰 비중을 차지하는 것은 Fig. 7과 같이 소프트웨어의 복잡한 구조이다.

소프트웨어 컴포넌트 설계 시 개별 컴포넌트 구조가 복잡하게 되면 분석이 어려워지고 설계나 구현에서 소프트웨어 개발자의 실수로 이어진다. 결국 시스템은 안전 요구사항을 만족하지 못하게 된다.

우리는 브레인스토밍을 통해 개발자의 실수가 발생되는 원인과 심각도의 관계를 고민하였다. 그 결과 분석대상 컴포넌트의 출력 신호와 후순위 컴포넌트에서 사용되는 입력 신호의 관계를 고려하여 심각도 기준을 만드는 것이 타당하다고 판단하였다. 소프트웨어 안전 분석 시 객관성을 가지고 분석할 수 있도록 심각도 기준을 Table 3과 같이 작성하였다. 기존에 사용하던 심각도 기준이 시스템 단계에서의 장애를 언급했다면, 변경된 표에서는 분석대상 컴포넌트의 기능 결함과 후순위 컴포넌트 기능 간의 관계에 대해서만 언급하였다.

Fig. 8에서 확인할 수 있듯이 제안 기법을 소프트웨어 FMEA에 사용하여 Fig. 6에서 식별된 고장 원인 “Input data trembling”이 삭제되었다. 또한 “Vehicle Speed Signal Handler” 컴포넌트에 존재하던 불필요한 안전 메커니즘 역시 삭제되었다. 즉 소프트웨어 영역에서 체계적이고 객관적인 분석이 가능하게 되면서 과다한 안전 메커니즘 문제가 개선된 것이다.

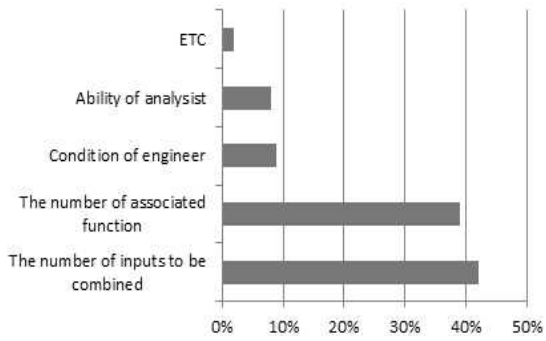


Fig. 7. Causes of Systematic Failure on Software Development.

Table 3. The Severity Table for Software FMEA

Severity	Proposed Scheme Software Severity Description
10	Direct violation of a complicated safety requirement(be implemented with three or more conditional statements) in the following software component using the function
9	Direct violation of a moderate safety requirement(be implemented with two conditional statements) in the following software component using the function
8	Direct violation of a straightforward safety requirement(be implemented with one conditional statements) in the following software component using the function
7	Violation of a following software component's safety requirement in combination with a other software component function
6	Violation of a following software component's safety requirement in combination with other software component functions(two)
5	Violation of a following software component's safety requirement in combination with other software component functions(three or more)
4	Affects non-safety requirements result of a following software component
3	Affects non-safety requirements result of a following software component with other software component functions(one)
2	Affects the non-safety requirements result of the following software component with other software component functions(two or more)
1	Does not affect the any requirements of the following software component

Effects	S	Failure mode	Cause	Preventive action	O
System element: Vehicle Speed Signal Handler					
Function: Sending speed information					
[System element 1] Does not affect the any requirements of the Mode Manager	1	Incorrect wheel speed	[System element 3] Systematic failure	Initial state: 2015-12-23 Preventive action 1 Revision state: 2017-11-29 -N/A	10 8
System element: Command Handler					
Function: Sending actuation command					
[System element 1] Unintended Apply request at high speed	8	Sending actuation command	Wrong Actuation enableline from Vehicle Status Detection and Wrong apply request from Mode Manager	Initial state: 2016-03-08 -[SM] Vehicle Status Detection	2

Fig. 8. Modified Software FMEA.

5. 평 가

기존에는 소프트웨어 FMEA 수행 시 하위 컴포넌트들을 동시에 분석 하였다. 각 컴포넌트의 개발자는 컴포넌트의 장애 유형과 시스템의 기능의 관계를 분석하여 컴포넌트별로 안전 메커니즘을 적용하였다. 분석결과에 따라 개발자가 적절하다고 판단하는 안전 메커니즘을 적용하였다. 각각의 안전 메커니즘은 해당 컴포넌트에 적절하게 적용되었다고 할 수 있었으나, 전체 시스템 수준에서는 과다하게 설계되어 문제를 일으켰다. 과다한 안전 메커니즘 때문에 설계단계에서 동료검토에 많은 시간이 사용되었다. 또한 과다한 안전 메커니즘은 테스트 단계에서 소프트웨어의 기능검증을 어렵게 하였고, 안전 메커니즘 검증에 투입되는 자원을 계획보다 많아지게 하였다.

4장에서는 소프트웨어의 분석, 변경, 검증 등, 유지보수의 어려움이 소프트웨어 설계측면의 심각도에 해당한다고 분석하였다. 그리고 유지보수의 어려움을 정량화하여 소프트웨어 FMEA 가이드라인을 제안하였다.

본 장에서는 제안된 가이드라인에 따라 소프트웨어 FMEA를 진행한 뒤 안전 분석의 효율성을 재평가한다. 제안 기법을 따르는 프로젝트의 소프트웨어 FMEA 결과와 시스템 FMEA 가이드라인을 따랐던 프로젝트의 소프트웨어 FMEA 결과를 비교하였다. 두 가지 프로젝트는 설계 인원을 제외하면 시스템

Table 4. The Rate of Change about Safety Requirements

The number of software safety requirements change rate	The number of technical safety requirements change rate
-13%	0%

기능과 안전 목표가 같다. 평가 대상은 Fig. 5에서 언급한 전자식 주차 브레이크 소프트웨어를 구성하는 5개 컴포넌트의 기능 및 도출된 안전 메커니즘에 대해서만 비교 분석한다.

먼저 Table 4와 같이 안전 메커니즘의 삭제 또는 변경이 적절한지 확인하기 위해 소프트웨어 안전 요구사항 개수와 시스템 단계의 기술적 안전 요구사항 개수의 변화량을 확인하였다. 소프트웨어 안전 요구사항 개수는 감소하였지만, 기술적 안전 요구사항 개수의 변화량은 없었다. 즉 소프트웨어 안전 요구사항의 감소 및 변경에도, 시스템의 안전 목표 충족에 대한 손실은 없다는 것을 확인하였다.

4.1 심각도 값의 재조정 및 객관성 확인

제안된 소프트웨어 FMEA 심각도 기준을 사용할 때 심각도 값의 변화를 확인한다. Table 5와 같이 기존에 사용하던 심각도 기준보다 제안된 심각도 기준을 사용할 때 전반적으로 심각도 값이 하향 조정된

Table 5. The Variation of Severity According to Methodology

Severity	Classic method	Proposed scheme
10-8	13	2
7-5	2	5
4-1	6	14

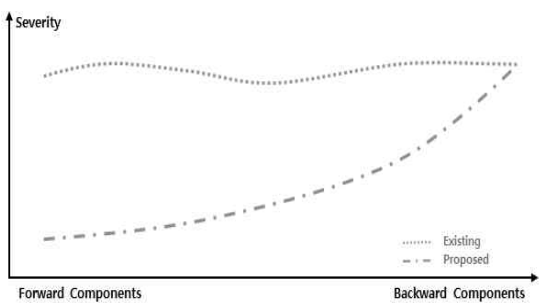


Fig. 9. The Relationship Between Component Location and Severity Value of Failure Mode.

것을 알 수 있다.

Fig. 9처럼 데이터의 흐름을 기준으로 앞쪽에 배치된 컴포넌트의 장애 유형에 대한 심각도 값이 후순위 컴포넌트 장애 유형에 대한 심각도 값보다 전반적으로 하향 조정되었다. 그 이유는 백-프런트의 분석을 진행하다 보니 후순위 컴포넌트가 먼저 안전 분석되었으며, 후순위 컴포넌트에 적용된 안전 메커니즘으로 앞쪽에 배치된 컴포넌트의 장애 유형에 대한 심각도 값이 낮아지게 된 것이다.

기존의 프로젝트에서는 동료검토 이후 심각도 값을 재조정하는 논의가 많았지만, 제안된 심각도 기준을 사용한 경우 심각도 값 재조정을 위한 논의는 없었다. 제안된 심각도 기준을 사용하면 대부분의 개발자가 장애 유형에 대한 심각도를 비슷한 값으로 판단하는 것을 확인하였다.

4.2 제안기법의 효율성

제안된 가이드라인을 적용한 프로젝트와 기존의 가이드라인을 따르는 프로젝트의 자원사용에 대한 효율성을 비교하였다. 효율성 비교는 변경되거나 삭제된 안전 메커니즘 개수로 하였다.

안전 분석이 수행된 이후 동료검토 또는 검증단계에서 안전 메커니즘의 적절함이 평가된다. 적절하지 못하다고 평가된 안전 메커니즘은 삭제 또는 변경되는데 그 개수는 불필요하게 사용된 자원의 양과 같다고 할 수 있다. Table 6과 7을 보면 안전 분석 후 변경되거나 삭제되는 안전 메커니즘이 기존의 방법에서 더 많다는 것이 확인된다. 즉 기존의 가이드라인을 소프트웨어 FMEA에 사용하기에는 적절하지 않다.

안전 메커니즘의 개수가 조정되면서 소프트웨어의 개선된 정도를 확인하기 위해 앞서 언급된 5개 컴포넌트의 실행시간을 측정하였다. 실행시간은 TRACE

Table 6. The Number of Changed Safety Mechanisms

Method	Changed safety mechanisms
Proposed Scheme	3 EA
Classic FMEA	47 EA

Table 7. The Number of deleted safety mechanisms

Method	Deleted safety mechanism
Proposed Scheme	0 EA
Classic FMEA	23 EA

Table 8. Changed Execution Time

Component name	Execution time	Change rate
Request Detection	67μs	-26%
Vehicle Speed Signal Handler	28μs	-42%
Vehicle Status Detection	41μs	0%
Mode Manager	17μs	-8%
Command Handler	103μs	0%

32 장비를 사용하여 측정하였으며 소프트웨어는 Infineon TC27X core, 16MB RAM의 환경에서 운용되었다.

Table 8과 같이 후순위 컴포넌트는 실행시간에 큰 변화가 없었으나, 먼저 수행되는 일부 컴포넌트에서는 실행시간 감소가 확인되었다. 과다했던 안전 메커니즘의 삭제로 개선된 것이다. 소프트웨어 개발 영역에서 가장 많이 삭제된 안전 메커니즘은 “입출력 값에 대한 범위검사”였다.

마지막으로 안전 분석의 반복횟수를 비교하였다. Table 9와 같이 제안된 가이드라인 적용 시 안전 분석의 수행횟수가 줄어드는 것을 알 수 있다. 기존 방법은 컴포넌트들의 안전 메커니즘이 추가 또는 삭제될 때마다 전체 안전 메커니즘에 대한 영향성이 달라졌고, 지속적인 비교 평가를 위해 안전 분석이 반복되었다.

제안된 가이드라인을 사용하면 분석 컴포넌트의 기능을 명확히 한 뒤 Fig. 10과 같이 백-프런트 순서로 안전 분석을 진행한다. 그 결과 장애 유형에 대한 후순위 컴포넌트의 영향성이 적절하게 식별되었으며, 안전 메커니즘이 적절하게 도출되면서 안전 메커니즘의 재평가 횟수가 줄어들었다. 안전 메커니즘의 재평가 횟수가 줄어들면서 소프트웨어 FMEA 횟수 역시 줄어들었다.

Table 9. The Number of Software FMEA Iterations

Method	Iteration times
Proposed Scheme	3
Classic FMEA	7

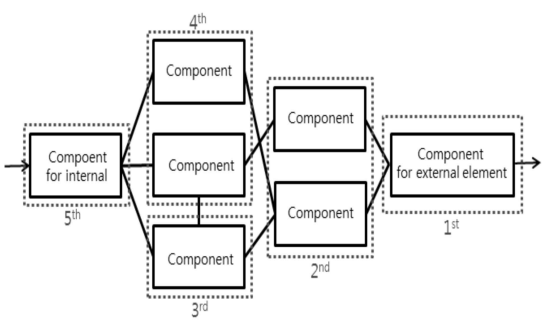


Fig. 10. The Analysis Order of Software Components.

평가를 통해 컴포넌트들을 순차적으로 안전 분석하는 것이 동시에 안전 분석하는 것보다 효과적임을 알 수 있었다. 백-프런트 순서로 분석하게 되면 컴포넌트의 분석 시점에 후순위 컴포넌트의 분석은 완료가 된다. 분석대상의 장애 유형과 영향성을 체계적으로 분석할 수 있게 되어, 안전 목표가 달성되는 최소한의 안전 메커니즘 설계가 더욱 용이해진다. 후순위로 수행되는 컴포넌트가 반드시 먼저 안전 분석이 완료되어야 하는 단점이 있지만, 전자식 주차 브레이크 시스템의 경우 3회의 안전 분석으로도 만족할 수 있는 결과가 도출되었다. 제안 가이드라인을 사용하면 소프트웨어 FMEA의 자원사용량과 분석 품질을 동시에 개선할 수 있게 된다.

6. 결 론

FMEA는 고위험 산업에서 오랜 기간 사용되어온 상향식 안전 분석 기법이며 효율성의 개선을 위해서 다양한 접근방법들이 연구되어 왔다. 하지만 소프트웨어 산업에서 FMEA가 연구되기 시작한 것은 비교적 최근에 진행되었다. 2000년대 초기에 P. L. Goddard, H. Pentti 등이 소프트웨어 FMEA에 대한 기법을 제시하였지만, 분석대상이 매우 작거나 SI 분야에 한정하였다. 해당 기법을 차량용 임베디드 소프트웨어 분야에 적용하기에는 분석의 어려움이 있었다[12, 13].

본 연구는 전자식 주차 브레이크 시스템 개발 시 소프트웨어 FMEA의 효율성을 향상시키기 위해 진행되었다. 소프트웨어 FMEA의 효율성을 높이기 위하여 본 논문에서 제시된 가이드라인의 주요 내용은

다음과 같다.

1. 소프트웨어 FMEA 분석대상이 되는 아키텍처 뷰
2. 안전 분석 시 고려해야 할 영향성 범위의 제한
3. 컴포넌트의 안전 분석 순서

기존의 가이드라인을 사용한 결과와 본 논문에서 제안한 가이드라인을 사용한 결과의 비교에서 제안한 가이드라인을 사용한 결과가 차량용 임베디드 소프트웨어 FMEA에 더욱 효율적인 것을 확인하였다. 안전 목표를 위한 요구사항이 13%, 소프트웨어 FMEA의 횟수가 50%이상 감소되는 것을 확인하였다. 안전 메커니즘의 조정으로 소프트웨어 실행시간도 개선되었다.

안전성 높은 자동차 시스템 개발을 위해서 임베디드 소프트웨어 FMEA 가이드라인을 제안하였으며, 제안된 가이드라인은 차량용 임베디드 소프트웨어에 적절하게 사용될 수 있다고 사료된다. 하지만 본 가이드라인을 타 소프트웨어 산업에서 사용하기에는 문제점이 있다. 산업별로 소프트웨어의 안전 목표가 다르기 때문이다. 또한 본 연구에서는 RPN 평가에 사용되는 발생도, 검출도 기준은 제안되지 못했다. 소프트웨어 영역에서 결함의 발견은 리뷰 등의 정적 분석 또는 테스팅 등의 동적 분석 행위가 전부이며, 이러한 소프트웨어의 특성상 발생도와 검출도를 정량적으로 측정하기는 매우 어려운 부분이기 때문이다.

앞으로 차량용 소프트웨어 산업은 ECU의 제공 기능의 다양화와 AUTOSAR 등의 다양한 플랫폼을 기반으로 빠르게 발전할 것이다. 따라서 소프트웨어 안전 분석 시 지금까지 고려하지 않았던 새로운 잠재적 장애 원인이 나타나게 될 것이다. 자동차 시스템을 연구하는 각 영역에서 소프트웨어 안전 분석의 중요성을 인식하고 영역 간 정보 교류 및 협조 속에 수준 높은 소프트웨어 안전 분석이 가능하기를 기대한다.

REFERENCE

[1] The International Organization for Standardization, *Functional Safety*, ISO 26262, Part 11, 2011.

[2] C. Price and N. Snooke, "An Automated Software FMEA," *Proceeding of International System Safety Regional Conference*, 2008.

[3] J. Catmur, M. Chudleigh, and F. Redmill, "Use of Hazard Analysis Techniques During the Product Life Cycle: HAZOP and FMEA Compared," *Proceedings of CSR 12 Annual Workshop on Safety and Reliability of Software Based Systems*, pp. 368-377, 1995.

[4] J.C. Knight and L.G. Nakano, "Software Test Techniques for System Fault-Tree Analysis", *The 16th International Conference on Computer Safety, Reliability and Security.* pp369-380, 1997.

[5] H. Yang, H.X. Wang, R.F. Han, and L. Juan, "Application of Fault Tree in Software Safety Analysis," *Proceedings of International Forum on Computer Science-Technology and Applications*, pp. 207-208, 2009.

[6] R. Souza and A.J. Alvares, "FMEA and FTA Analysis for Application of the Reliability Centered Maintenance Methodology: Case Study on Hydraulic Turbines," *Proceeding of ABCM Symposium Series in Mechatronic*, Vol. 3, pp. 803-812, 2008.

[7] Z. Hong and L. Binbin, "Integrated Analysis of Software FMEA and FTA," *Proceedings of International Conference on Information Technology and Computer Science*, pp. 184-187, 2009.

[8] MIL-STD 1629, *Procedures for Performing a FMEA and Effect Analysis*, 1980.

[9] SAE ARP 5580, *Recommended Failure modes and Effects Analysis (FMEA) Practices for Non-Automobile Applications*, 2001.

[10] SAE J1739, *Potential Failure Mode and Effects Analysis in Design (Design FMEA), Potential Failure Mode and Effects Analysis in Manufacturing and Assembly Processes*, 2009.

[11] VDA-Vol. 4, *Product and Process FMEA*, Germany, 1996.

[12] P.L. Goddard, "Software FMEA Techniques," *Proceedings of Reliability and Maintainabil-*

ity Symposium, pp. 118-123, 2000.

[13] H. Pentti and H. Atte, "Failure Mode and Effects Analysis of Software-Based Automation System," *Stuk-yto-tr 190,* pp. 1-37, 2002.

[14] M.H. Kim and M.G. Kim, "A Study on the Software Fault Modes and Effect Analysis for Software Safety Evaluation," *Journal of Korea Multimedia Society*, Vol. 15, No. 1, pp115-130, 2012.

[15] B. Ward, "A Demonstration to Assess Effectiveness, Suitability, and Survivability With the Missions and Means Framework," *Army Reearch Laboratory*, pp1-156, Dec 2012.

[16] M. Choi, J. Kim and J. Lee, "On Enhancing Safety of Train-Centric Train Control System using Model-Based Development," *Journal of the Korea Academia-Industrial cooperation Society*, Vol17. No. 7, pp.573-584, 2016.

[17] Maier "FMEA and FTA to Support Safe Design of Embedded Software in Safety-Critical Systems" *Springer Safety and Reliability of Software Based Systems*, pp. 351-356, 1997.

[18] C.S. Carlson, "Understanding and Applying the Fundamentals of FMEAs," *Proceeding of 2015 Annual Reliability and Maintainability Symposium*, pp. 1-32, 2014.

[19] P.B. Kruchten, "The 4+1 View Model of architecture," *IEEE Software*, Vol. 12, Issue 6, pp. 42-50, 1995.

[20] P. Clements, D. Garlan, R. Little, R. Nord, J. Stafford., "Documenting Software Architectures: Views and Beyond", *Proceedings of the 25th International Conference on Software Engineering*, pp740-741, 2003.

최 준 열

2012년 성균관대학교 전자전기컴퓨터(석사)
2013년~현재 경북대학교 컴퓨터학부 박사과정
2012년~2015년 한국항공 헬기항전SW팀 연구원
현재 만도 Global R&D 제동센터 설계1팀 연구원
관심분야 : 차량 시스템 안전 설계, 차량 시스템 검증, 차량 샤시 제어

김 용 길

2002년 연세대학교 기계공학과 (학사)
2003년~2016년 만도 Global R&D 제동센터 SW개발팀 연구원
현재 만도 Global R&D 제동센터 설계1팀 파트장
관심분야 : 차량 샤시 제어, 인공지능

조 준 형

1996년 경북대학교 전자공학과 (석사)
2010년~2015년 만도 Global Quality 전자품질팀 팀장
현재 만도 Global R&D 제동센터 설계1팀 팀장
관심분야 : 차량 샤시 품질

최 윤 자

2003년 미네소타대학교 전산과 (박사)
2003년~2006년 독일 프라운호퍼 연구소 연구원
현재 경북대학교 컴퓨터학부 교수
관심분야 : 소프트웨어 안전성 분석, 정형검증, 모델기반 개발방법론