공학석사 학위논문

# 소프트웨어 신뢰성 향상을 위한 AUTOSAR의 개선된 안전 메커니즘

Improved Safety Mechanisms of AUTOSAR for Enhancing Software Reliability

2016 년 2 월

서울대학교 대학원

전기·정보공학부

윤 현 준

# 소프트웨어 신뢰성 향상을 위한 AUTOSAR의 개선된 안전 메커니즘

Improved Safety Mechanisms of AUTOSAR for Enhancing Software Reliability

지도 교수 홍 성 수

이 논문을 공학석사 학위논문으로 제출함
2016 년 1 월

서울대학교 대학원
전기·정보공학부
윤 현 준

윤현준의 공학석사 학위논문을 인준함
2016 년 1 월

위 원 장 김 태 환 (인)

부위원장 홍 성 수 (인)

위 원 문 수 묵 (인)

# 초 록

최근 무인자율주행 자동차 기술 개발을 위해 자동차 업계의 연구가 활발히 진행되고 있다. 그 결과 주요 자동차 제조 회사들은 고급 차종에 첨단 운전자 보조 시스템을 탑재하고 있다. 이를 위해 각종 센서와 ECU와 같은 전자장치의 비중이 증가됨에 따라 차량용 소프트웨어의 크기와 복잡성이 증가하고 있다. 이러한 상황에 대응하기 위해 유럽의 주요 자동차 제조 회사를 필두로 AUTOSAR라는 표준화된 차량용 소프트웨어 플랫폼을 만들었다. 또한, 갈수록 증가하는 차량용 전기/전자 시스템의 오작동으로 인한 사고방지를 위해 ISO 26262라는 자동차 기능 안전 국제 규격이 등장하였다. AUTOSAR는 ISO 26262의 기능 안전 요구사항을 만족시키기 위해 safety mechanism을 도입하였다. 하지만 현재 최신 버전의 AUTOSAR에 존재하는 safety mechanism은 ISO 26262가 정의한 소프트웨어 fault들을 완벽히 다루지 못한다. 먼저, 필요 시 AUTOSAR 소프트웨어의 올바른 수행을 확인하는 program flow monitoring이 특정 상황에서 오류를 검출해 내지 못한다. 그리고 응용 소프트웨어 간 올바른 데이터 전송을 확인하는 E2E protection이 데이터의 신선함을 확인하는 기능을 가이드라인만 제시할 뿐 표준 API를 제공하지 않는다. 따라서 본 논문에서는 ISO 26262의 소프트웨어 fault들을 완벽히 다루는 AUTOSAR의 safety mechanism 개선 방안을 제안한다. 첫째, AUTOSAR 소프트웨어의 모든 상황에서 올바른 수행을 확인하도록 개선된 program flow monitoring을 제안한다. 둘째, E2E protection의 전송된 데이터의 신선함을 확인하는 표준 API를 제안한다. 본 논문에서는 개선한 AUTOSAR의 safety

mechanism을 실험을 통해 검증하였다.

**주요어** : AUTOSAR, 차량용 소프트웨어 플랫폼, Safety Mechanism, Program Flow Monitoring, E2E Protection
**학 번** : 2014-21633

# 목 차

# 표 목차



# 그림 목차

# 제 1 장 서 론

자율주행 기능을 탑재한 자동차를 개발하기 위한 전세계 자동차 제조 회사 간의 경쟁이 치열하다. 각 자동차 제조 회사 마다 완전한 자율주행 기술 개발을 위해 여러 단계에 걸친 주행 자동화 목표를 설정한 로드맵을 구상하고 있다. BMW와 Continental은 2025년 완전한 자율주행 자동차를 목표로 1단계는 2016년까지 부분적 자동화 기술 개발, 2단계는 2020년까지 높은 자동화 기술 개발, 마지막 3단계는 2025년까지 완전한 자동화 기술 개발을 완료할 것이라 발표했다[1]. 부분적 자동화는 비록 차량이 매우 제한된 상황에서 주행 기능을 담당하지만 운전자는 항상 차량의 주행 상황을 확인하고 필요하면 언제든 차량 제어를 할 수 있어야 한다. 높은 자동화는 부분적 자동화와 마찬가지로 차량이 제한된 상황에서만 주행 기능을 담당하지만 운전자가 일정 시간 동안 차량 주행 상황을 주시할 필요가 없다. 완전한 자동화는 차량이 출발 지점부터 도착 지점까지 운전자의 개입 없이 능동적으로 주행 기능을 제공하는 것이다. 현재 주요 자동차 제조 회사들은 고급 차종을 대상으로 자율주행 기술 실현의 바탕이 되는 첨단운전자보조시스템(Advanced Driver Assistance System, ADAS)을 장착하고 있다. 첨단운전자보조시스템을 위해선 각종 센서와 센서들이 만들어 낸 데이터를 처리하고 차량을 제어하기 위한 전자제어장치(Electronic Control Unit, ECU)가 다수 필요하다. 또한, 이들은 CAN, FlexRay, LIN과 같은 차량용 네트워크를 통해 연결되어

있다. 차량 내 존재하는 ECU는 각기 다른 기능을 제공하며 그 기능에 맞춰 하드웨어 구성요소와 특성 역시 달라진다. 이에 맞춰 자동차 부품 업체들은 하드웨어 특성에 맞는 소프트웨어를 개발하였으며, ECU의 하드웨어 구성이 달라지면 다시 소프트웨어 개발을 해야만 했다. 또한, 자동차 제조 회사에서 다양한 부품회사로부터 제공받은 ECU들을 차량 내 탑재하여 통합하는 과정에서 많은 시간이 소요되었다. 이러한 문제점을 해결하고자 2003년 BMW, Daimler AG, Continental, Bosch 등의 자동차 산업 관련 업체들이 모여 컨소시엄을 형성하였고 AUTOSAR라는 차량용 소프트웨어 플랫폼 표준을 개발하였다. AUTOSAR는 차량용 소프트웨어를 계층적 구조를 만들어 ECU, 네트워크 등의 하드웨어 구조와 독립적으로 차량용 소프트웨어를 개발 가능하게 하며, 소프트웨어의 크기와 복잡성을 효과적으로 관리할 수 있도록 한다. 또한, 소프트웨어를 모듈화하여 소프트웨어의 재사용성을 높이고 개발 기간을 단축하도록 한다[2]. AUTOSAR는 2006년 버전 2.1에서 AUTOSAR를 구성하는 주요 표준들이 완성되었으며, 버전 4.0 이후, 멀티코어 지원과 새로운 모듈 다수 추가하였고 계속된 유지보수 및 개선을 통해 2015년 현재 AUTOSAR는 4.2 버전이 공개되어 있다[3].

한편, 갈수록 증가하는 차량 내 전자장치의 안전성의 중요해지고 차량용 전기/전자 시스템의 복잡도 증가로 인한 개발 프로세스의 표준화 필요성이 증가됨에 따라 차량용 전기/전자 시스템의 기능 안전 표준인 ISO 26262가 제정되었다. ISO 26262는 차량 내 탑재되는 전기/전자 시스템의 기능 안전 개념부터 제품 개발, 생산 및 운영, 이를 위한 지원

과정 등을 명시한 표준이다.

AUTOSAR는 ISO 26262가 제시한 기능 안전 요구사항을 만족시키기 위해 차량 내 발생 가능한 모든 소프트웨어 fault들의 예측, 방지, 감내, 제거의 방법을 이용하여 소프트웨어의 신뢰성을 높이는 safety mechanism을 도입한다.

하지만 현재 최신 버전의 AUTOSAR의 safety mechanism의 기능 중 일부는 ISO 26262에서 정의한 소프트웨어 fault 중 일부를 완벽히 다루지 못한다. 첫 번째로, 필요 시, 소프트웨어의 수행이 올바른지 확인하는 기능을 담당하는 program flow monitoring에서 deadline supervision 기능에 한계점이 있다. Deadline supervision이란 소프트웨어의 특정 구간이 정해진 시간 내에 수행이 되는지 체크하는 기능이다. 소프트웨어 내 측정 구간을 설정하기 위해 두 개의 체크포인트가 사용되며, 첫 번째 체크 포인트를 CheckPointStart라 하며, 두 번째 체크 포인트를 CheckPointEnd라 한다. Deadline supervision을 사용 중, 수행 확인 대상이 되는 소프트웨어가 CheckPointStart를 지나고 CheckPointEnd를 통과하기 전 중간에 수행이 중단이 되면 이 상황을 오류로 인식하지 않는다. 두 번째로, 응용 소프트웨어 간 전송되는 데이터가 올바른지 확인하기 위한 기능을 제공하는 E2E protection에서 timeout detection 기능에 한계점이 있다. Timeout detection이란 데이터를 수신하는 응용 소프트웨어가 현재 수신한 데이터의 신선도를 확인하는 기능이다. 즉, 응용 소프트웨어가 데이터를 수신하였을 때, 현재 수신한 데이터가 이전에 수신된 데이터와 얼마나 시간 간격이 벌어지는 지를 체크하여 정해진 시간보다 늦게

도착한 경우 현재 도착한 데이터가 신선하지 않다고 판단한다. 하지만 이러한 전송된 데이터의 신선도를 확인하는 기능을 제공하는 freshness API는 AUTOSAR에서 개발자가 직접 구현하도록 되어있다. 따라서 차량용 소프트웨어 업체들이 각기 다른 방식으로 freshness API를 개발하게 된다. 이는 곧 freshness API 개발 상에서 오류를 제공할 수 있고 자동차 업체들은 같은 기능을 제공하지만 사용 방식이 다른 API들을 사용함에 따른 오류 발생을 야기한다. 뿐만 아니라, 서로 연결하기 위한 추가적인 작업이 필요하고 이는 AUTOSAR의 개발 동기인 소프트웨어의 재사용성과 개발기간 단축에 반하는 것이다.

본 논문은 위에서 언급한 AUTOSAR의 safety mechanism의 한계점을 개선하여 소프트웨어 신뢰성을 향상시키는 방안을 제안한다. 먼저 program flow monitoring의 deadline supervision이 수행 확인 대상인 소프트웨어가 CheckPointStart를 통과하고 CheckPointEnd를 지나기 전 수행이 중단 되어도 주기적으로 CheckPointStart를 지난 시점과 현재 시각을 비교하여 정해진 시간을 넘어간 경우 오류로 판단할 수 있도록 한다. 그리고 E2E protection의 응용 소프트웨어 간 전송되는 데이터의 신선도 확인용 표준 API를 구현하고 이것에 대한 가이드라인을 제시하여 표준 API 부재로 인한 문제를 해결한다. 실험을 통하여 제안한 기법의 실효성을 검증하였다.

본 논문의 나머지는 부분은 다음과 같이 구성된다. 2장은 논문의 이해를 돕기 위해 AUTOSAR와 AUTOSAR의 safety mechanism에 대한 배경지식을 설명하고 3장은 현재 최신 버전의 AUTOSAR가 제공하는 safety mechanism의 한계점을 구체적으로 설명한다.

4장에서는 제안하는 해결책을 설명하고, 5장에서는 실험을 통한 검증을 수행한다. 6장에서는 논문의 결론을 맺는다.

# 제 2 장 배 경

이 장에서는 본 논문의 이해를 돕기 위한 배경 지식을 설명한다. 2.1절에서는 AUTOSAR에 대해 설명하고, 2.2절에서는 ISO 26262의 소프트웨어 개발 부분을 설명한다. 마지막으로 2.3절에는 ISO 26262에 맞춰 AUTOSAR가 변화된 점을 설명한다.

## 2.1 AUTOSAR

AUTOSAR는 자동차 제조 회사, 자동차 부품 회사, 차량용 소프트웨어 회사와 반도체 회사 등 자동차 산업과 관련한 업체가 참여하여 만든 차량용 소프트웨어 플랫폼 표준이다. AUTOSAR를 이루는 표준은 그림 1과 같이 세 가지로 나눌 수 있다. 첫째, 기본 소프트웨어(Basic Software, BSW)와 런타임 환경(Run-Time Environment, RTE)에 대한 명세를 포함하는 소프트웨어 아키텍처(Software Architecture), 둘째, 소프트웨어 개발 방법론과 각 개발 방법에 사용되는 입력 문서와 출력 문서의 양식을 정의한 방법론과 템플릿(Methodology and Templates), 다양한 차량용 응용 소프트웨어를 AUTOSAR 기반으로 구현 시, 참조 모델을 정의한 응용 인터페이스(Application Interfaces)가 있다.

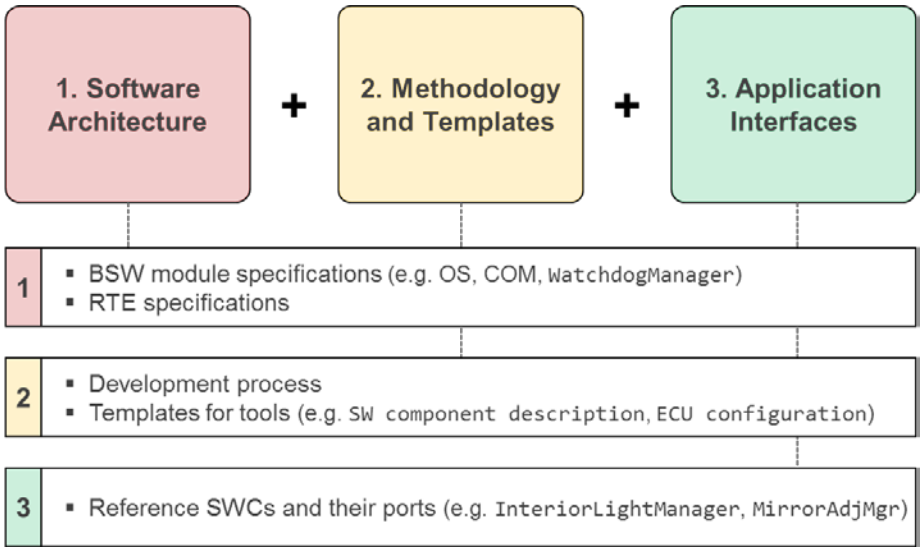


그림 1. AUTOSAR 표준의 구성 [4]

AUTOSAR의 소프트웨어 아키텍처는 그림 2처럼 상위에서부터 하위로 다음과 같이 세 가지로 분류 된다.

- 응용 계층(Application Layer);
- 런타임 환경(Run-Time Environment);
- 기본 소프트웨어(Basic Software).

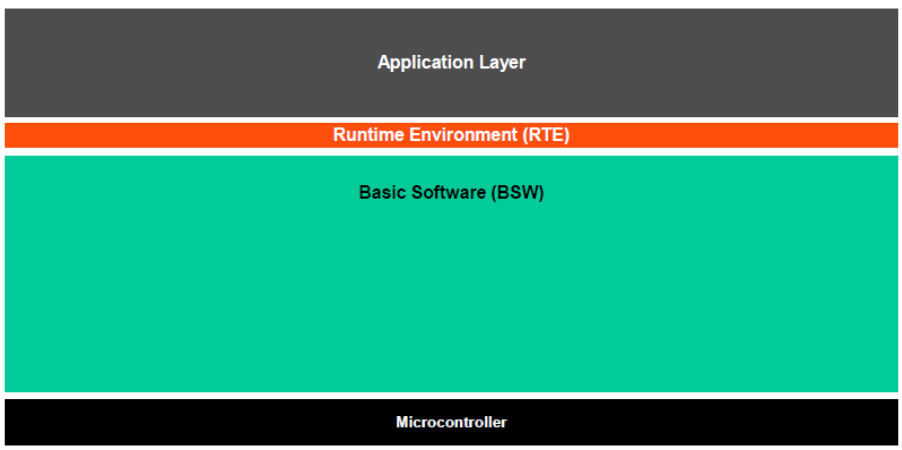


그림 2. AUTOSAR의 계층적 구조 [2]

응용 계층은 해당 하드웨어가 차량 내 어떤 기능을 할 지를 결정한다. 응용 계층에 존재하는 소프트웨어는 재사용이 가능한 최소 기능 단위인 소프트웨어 컴포넌트(Software Component, SWC)로 구성된다. 차량 내 전기/전자 시스템 구성을 위한 소프트웨어 컴포넌트들 설계 시, 추상화된 가상의 네트워크인 가상 기능 버스(Virtual Functional Bus, VFB)를 통해 기반 하드웨어에 독립적으로 소프트웨어 컴포넌트를 설계할 수 있다. 또한, AUTOSAR는 미리 정의된 인터페이스를 제공하며, 이를 이용하여 소프트웨어 컴포넌트 간 혹은 소프트웨어 컴포넌트와 BSW 모듈 간 통신을 하게 된다. 모든 소프트웨어 컴포넌트는 오직 포트를 통해서만 데이터를 내보내고 받을 수 있다.

AUTOSAR에서 미리 정의한 인터페이스는 세 가지로 분류된다.

- AUTOSAR Interface;
- Standardized AUTOSAR Interface;
- Standardized Interface.

AUTOSAR interface는 소프트웨어 컴포넌트 간 혹은 소프트웨어 컴포넌트와 BSW 모듈 간 교환되는 정보를 정의한 것으로 프로그래밍 언어, 하드웨어 구성과는 무관하다. RTE를 통해서 제공된다. Standardized AUTOSAR interface는 BSW 모듈이 제공하는 다양한 서비스들을 정의한다. Standardized interface는 AUTOSAR interface를 사용하지 않고 AUTOSAR 내에서 표준화 된 API들을 뜻한다. 이 interface들은 보통 특정 언어로 정의가 되어있다.

소프트웨어 컴포넌트는 하나 이상의 러너블(Runnable)로 구성되며

런타임에 독립적으로 스케줄링되고 수행되는 단위이다.

RTE는 위 계층인 응용 계층에 존재하는 응용 소프트웨어 간 통신 서비스를 제공한다. 앞서 설명한 VFB의 구현 형태가 바로 RTE이다. 이 계층을 이용하여 응용 소프트웨어는 통신하고자 하는 응용 소프트웨어가 어떤 ECU 상에 존재하는지 고려할 필요 없이 통신할 수 있도록 해준다. RTE를 상위 계층인 응용 계층은 계층적 구조에서 컴포넌트 스타일로 소프트웨어 아키텍처 스타일이 바뀐다.

BSW는 하드웨어 바로 위에 위치하며 하드웨어 자원을 사용하고 다양한 시스템 기능을 제공한다. BSW 역할에 따라 마이크로컨트롤러 추상 계층(Microcontroller Abstraction Layer, MCAL), ECU 추상 계층(ECU Abstraction Layer), 복합 장치 드라이버(Complex Device Drivers), 서비스 계층(Service Layer)의 세부 계층으로 나뉘어 지며 그림 3과 같다.

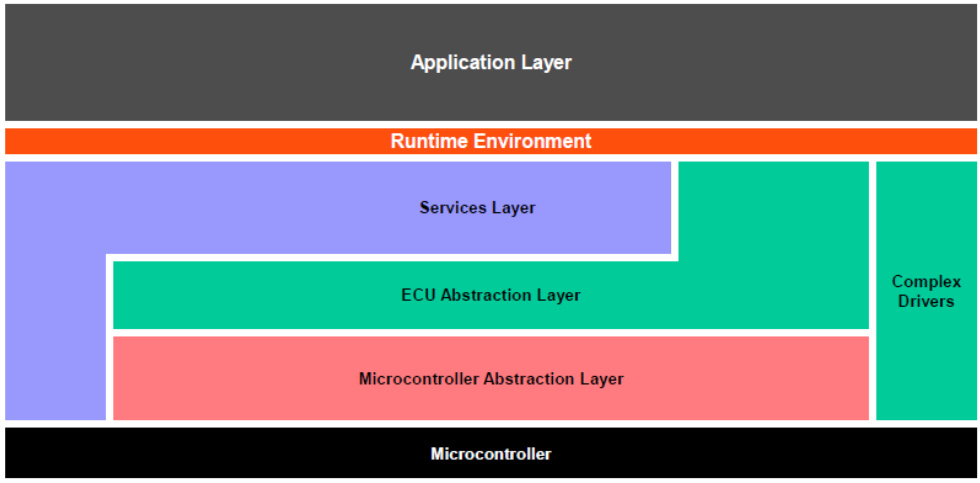


그림 3. AUTOSAR의 계층적 구조: 기본 소프트웨어 상세 [2]

BSW의 각 계층에 대한 설명은 다음과 같다. 먼저 마이크로컨트롤러 추상 계층은 BSW의 가장 아래에 위치하며, 마이크로컨트롤러와 내부 주변 기기들에 대한 직접 접근을 하게 해주는 내부 드라이버를 포함한다. 이 계층은 위 소프트웨어 계층을 마이크로컨트롤러와 독립적으로 만든다. ECU 추상 계층은 MCAL 위에 존재하며, 외부 플래시 메모리와 외부 워치독 같은 외부 기기들에 대한 접근을 하게 해주는 드라이버를 포함한다. 또한, 각종 장치에 대한 표준화된 API를 제공한다. 이 계층은 위 소프트웨어 계층을 ECU 하드웨어와 독립적으로 만들어 준다. 복합 장치 드라이버는 특수한 목적을 가진 기능을 담당하는 소프트웨어 개체로 BSW 모듈에 접근할 수 있다. 또한, AUTOSAR에서 정의되지 않은 마이크로컨트롤러 인터페이스를 통합할 수 있다. 서비스 계층은 BSW의 가장 최상위 계층이며, 태스크 스케줄링과 같은 운영체제 기능, 네트워크 통신과 관리 서비스, 진단 서비스, ECU 상태 관리 등의 기능을 제공한다.

BSW의 세부 계층들은 그 특성에 따라 구분된다. 먼저, MCAL은 마이크로컨트롤러 드라이버, 메모리 드리아버, 통신 드라이버, 입출력 드라이버로 구성된다. ECU 추상 계층은 OnBoard 장치 추상, 메모리 하드웨어 추상, 통신 하드웨어 추상, 입출력 하드웨어 추상으로 구성된다. 서비스 계층은 시스템 서비스, 메모리 서비스, 통신 서비스로 구성된다. BSW의 세부 계층들의 기능들은 소프트웨어 모듈로 구성되어 있으며 이를 BSW 모듈이라 한다. 예를 들어, MCAL의 통신 드라이버의 경우, 각각의 통신 기술에 대한 모듈로 구성된다. 즉, LIN 드라이버 모듈, CAN 드라이버 모듈, FlexRay 드라이버, Ethernet 드라이버

등으로 구성된다.

## 2.2 ISO 26262

이 절에서는 본 논문에서 제안하고자 하는 주제와 관련 있는 ISO 26262의 표준 중 소프트웨어 개발 관련 부분에 대한 배경 지식을 설명한다. ISO 26262는 자동차의 기능을 담당하는 하나의 전기/전자 시스템을 대상으로 적용되며, 그림 4와 같이 시스템 개발 시, V모델을 통해 시스템 수준의 개발을 거쳐 하드웨어 수준의 개발, 그리고 소프트웨어 수준 개발을 하게 된다. 시스템 수준 개발, 하드웨어 수준 개발, 소프트웨어 수준 개발 각각 설계 단계, 검증 및 평가 단계가 존재한다.

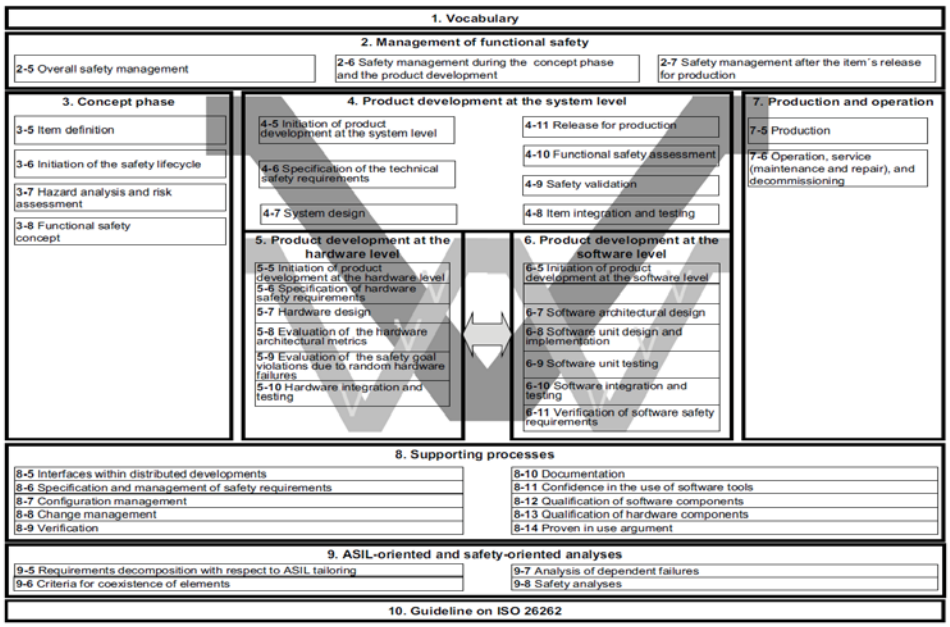


그림 4. ISO 26262 개관 [5]

ISO 26262에서 같이 차량용 소프트웨어 상에서 발생 가능한 fault들을 아래와 같이 세 가지로 분류하고 있으며, 그림 5와 같이 각 분류의 세부 fault들을 정의하고 있다[5].

- 시간과 수행(Timing and Execution);
- 메모리(Memory);
- 정보 교환(Exchange of Information).

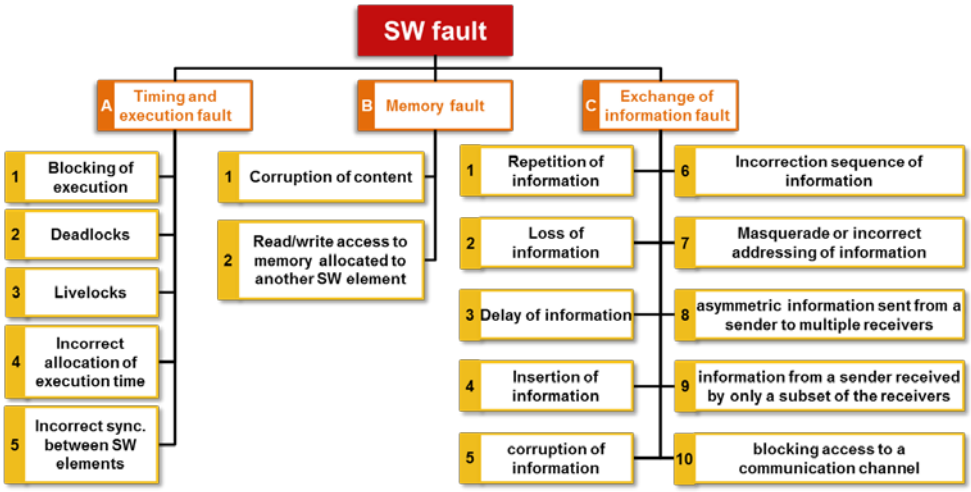


그림 5. ISO 26262에서 정의한 소프트웨어 fault 분류 [5]

## 2.3 AUTOSAR의 기능 안전 고려

2003년 AUTOSAR가 개발 시작된 이후, 2006년 버전 2.1을 통해 AUTOSAR 소프트웨어 아키텍처와 개발 방법론 등 표준의 주요 구성 요소를 완성했다. 이후, 3.x 버전을 통해 AUTOSAR의 표준이 보완되었으며, 2009년, 4.0 버전에서 다양한 네트워크 기술을 지원하는 통신 스택이 추가되고 멀티 코어 환경을 지원하기 위한 BSW 모듈의

추가와 보완이 이루어 졌다. 2011년 ISO 26262가 제정된 이후, AUTOSAR에도 기능 안전 요구사항이 적용되었고 이에 대응하고자, 2011년 12월 버전 4.0.3에서 처음으로 safety mechanism이라는 개념이 도입되었다[6]. AUTOSAR에는 이미 소프트웨어에서 발생 가능한 fault와 오류를 막기 위한 메커니즘이 모듈별로 존재하였지만, Safety mechanism이라는 개념이 정립되고 소프트웨어 상에서 발생 가능한 fault들에 대응하는 모든 메커니즘을 도입한 것은 4.0.3이 최초이다. Safety mechanism이란 하나의 독립된 BSW 모듈이 아닌 BSW 모듈들 내에 산재해 있으며 소프트웨어 상에서 발생 가능한 fault들을 예측, 방지, 감내, 제거의 방법을 통하여 fault가 고장으로 이어지지 않게 차단한다.

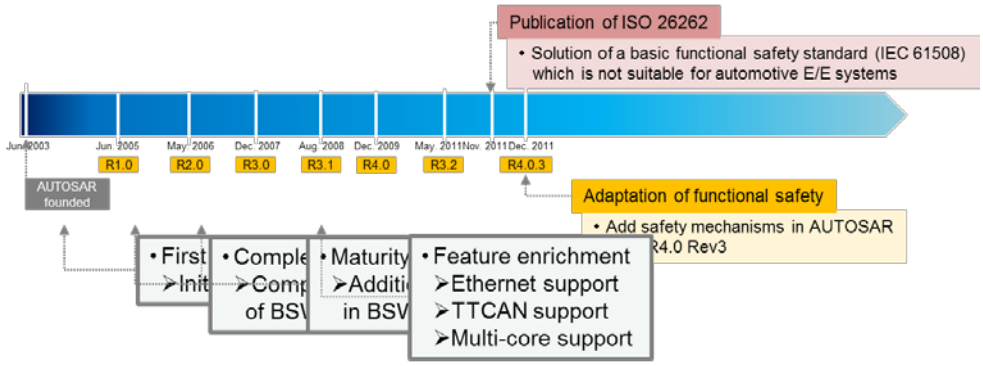


그림 6. ISO 26262와 관계를 고려한 AUTOSAR 개발 역사

## 2.4 AUTOSAR의 Safety Mechanism

현재 4.2 버전 AUTOSAR에는 크게 5가지 safety mechanism이 존재한다.

- 프로그램 흐름 모니터링(Program Flow Monitoring)
- E2E 보호(E2E Protection)
- 메모리 분할(Memory Partitioning)
- 통신 스택 관련 기능(Communication Stack Related Feature)
- 시간 동기화(Time Synchronization)

이후 이어질 이 절의 내용은 5 가지 safety mechanism에 대하여 설명한다.

■ 프로그램 흐름 모니터링

프로그램 흐름 모니터링은 소프트웨어 컴포넌트와 BSW 모듈을 대상으로 모니터링 대상이 올바른 수행이 되고 있는지 런타임에 확인하는 기능이다. 프로그램 흐름 모니터링의 수행은 워치독 매니저(Watchdog Manager, WdgM)가 담당한다. WdgM은 BSW 모듈 중 하나로써 서비스 계층의 시스템 서비스 부분에 속한다. WdgM의 역할은 크게 두 가지다. 첫째, 하드웨어 워치독이 시스템을 리셋시키지 않도록 주기적으로 하드웨어 워치독을 트리거한다. 둘째, 프로그램 흐름 모니터링을 수행한다. WdgM이 제공하는 프로그램 흐름 모니터링의 기능은 세 가지로 분류할 수 있다.

- Alive Supervision;
- Deadline Supervision;
- Logical Supervision.

각 세 기능 수행 시, 모니터링의 단위는 supervision entity (SE)를 사용한다. SE에 대한 명확한 기준은 AUTOSAR에 명시되어 있지 않으며 일반적으로 하나의 소프트웨어 컴포넌트, 하나의 BSW 모듈 또는 소프트웨어 컴포넌트 내 하나의 러너블이 된다.

Alive supervision은 주기적으로 수행되는 SE에 사용되며, 이 SE가 특정 시간 간격 동안 수행된 횟수를 측정하여 이 값이 미리 정해 놓은 최소 횟수와 최대 횟수 내에 있는지 확인한다. 만약, 최소 횟수와 최대 횟수 사이에 측정 값이 존재 한다면 정상 상태로, 그렇지 않다면 비정상 상태로 인식한다. Alive supervision을 사용하는 SE 마다 시간 간격, 최소 수행 횟수와 최대 수행 횟수를 개발 단계에 미리 설정한다. Alive supervision의 구체적 동작 원리는 그림 7과 같다.

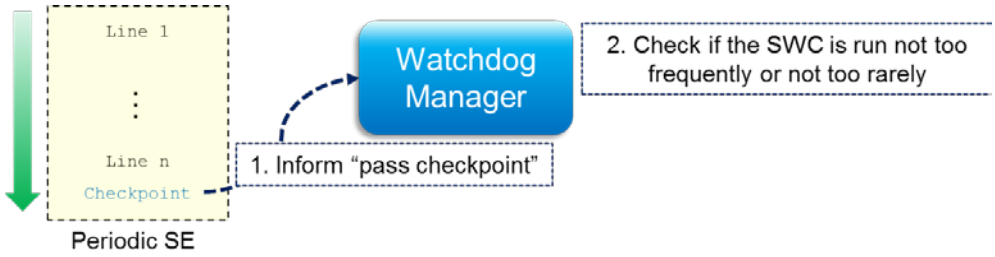


그림 7. Alive Supervision 메커니즘

Alive supervision을 사용하고자 하는 SE는 하나의 체크 포인트를 가지며 이 체크 포인트를 지날 때마다 WdgM에게 SE가 수행됨을

알리고 WdgM는 SE로부터 수행 알림을 받을 때마다 현재 수행 횟수를 카운트하고 특정 시간 간격 마다 수행 횟수가 최대 수행 횟수와 최소 수행 횟수 내에 있는지 확인한 뒤, 정상 범위 내 라면 SE의 수행 횟수를 초기화하고, 정상 범위가 아니라면 오류 처리 작업을 하게 된다.

Deadline supervision은 비주기적으로 수행되는 SE에 사용되며, 이 SE가 특정 코드 구간을 수행한 시간이 미리 정해 놓은 최소 수행 시간과 최대 수행 시간 내에 있는지 확인한다. 만약, 최소 수행 시간과 최대 수행 시간 사이에 측정 값이 존재 한다면 정상 상태로, 그렇지 않다면 비정상 상태로 인식한다. Alive supervision을 사용하는 SE 마다 측정 해야 할 구간, 최소 수행 시간과 최대 수행 시간을 개발 단계에 미리 설정한다. Deadline supervision의 구체적 동작원리는 그림 8과 같다. Deadline supervision을 사용하고자 하는 SE는 CheckPointStart와 CheckPointEnd라는 두 개의 체크 포인트를 가지며 SE는 먼저 CheckPointStart를 지날 때, 시각을 측정하여 WdgM에게 전달한다. 이후, CheckPointEnd를 지날 때, 현재 시각과 앞서 측정한 CheckPointStart을 통과한 시각의 차이를 계산하여 WdgM에게 전달한다. WdgM은 이 시각 차이 값을 미래 정해 놓은 최소 수행 시간과 최대 수행 시간 내에 있는지 확인한 뒤, 정상 범위 내 라면 SE의 두 체크 포인트에 대한 시각을 초기화하고, 정상 범위가 아니라면 오류 처리 작업을 하게 된다.

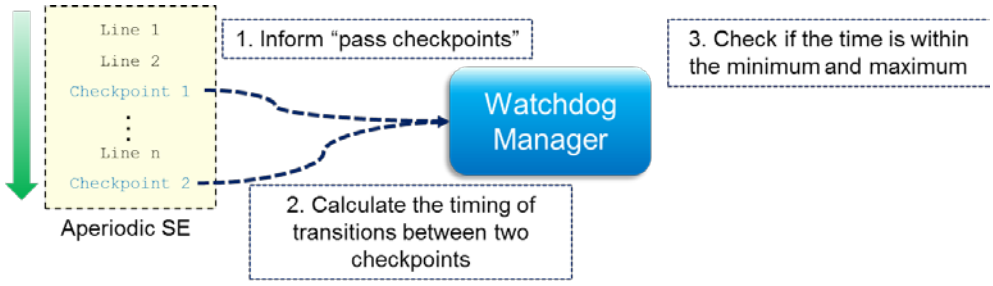


그림 8. Deadline Supervision 메커니즘

Logical supervision은 SE의 제어 흐름을 기반으로 논리적으로 타당한 수행 순서를 지키는지 확인하며, 이를 위해 다수의 체크 포인트를 사용한다. 또한, 이들 체크 포인트를 기준으로 각 체크 포인트마다 어떤 체크 포인트로 분기될 수 있는지를 파악하도록 제어 흐름 그래프를 작성한다. 이를 바탕으로 SE의 각 체크 포인트마다 해당 체크 포인트로 도달할 수 있는 체크 포인트 리스트와 다음으로 수행할 수 있는 체크 포인트 리스트가 존재하게 되고 이 리스트를 바탕으로 런타임에 SE가 논리적으로 타당한 수행이 되고 있는지 확인한다.

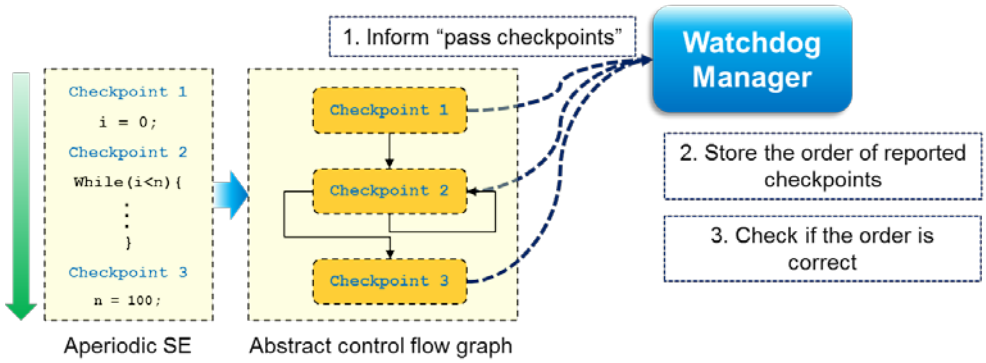


그림 9. Logical Supervision 메커니즘

각 체크 포인트, 제어 흐름 그래프와 체크 포인트의 리스트는 개발 단계에 정의된다. Logical supervision의 구체적 동작원리는 그림 9와 같다. Logical supervision을 사용하고자 하는 SE는 다수의 체크 포인트를 가지며, 각 체크 포인트마다 해당 체크 포인트에 도달할 수 있는 이전 체크 포인트들과 해당 체크 포인트가 다음으로 도달할 수 있는 이후 체크 포인트들로 나뉜다.

이러한 정보는 WdgM에 저장되어 있으며, SE가 매 체크 포인트를 지날 때마다 WdgM에게 알리고 WdgM은 해당 체크 포인트가 이전에 SE가 알려온 체크 포인트로부터 도달할 수 있는 체크 포인트인지 타당성을 확인한다. 만약 정상적인 도달이 아니라면 오류 처리 작업을 하게 된다.

프로그램 흐름 모니터링의 모든 기능의 결과는 정상 상태와 비정상 상태로 나뉘며, 정상 상태일 경우에 모니터링을 계속 수행한다. 비정상 상태일 경우, 그림 10와 같이 5가지의 오류 처리 작업을 수행할 수 있으며, 개발 단계에서 어떤 오류 처리 작업을 수행할지 결정한다.

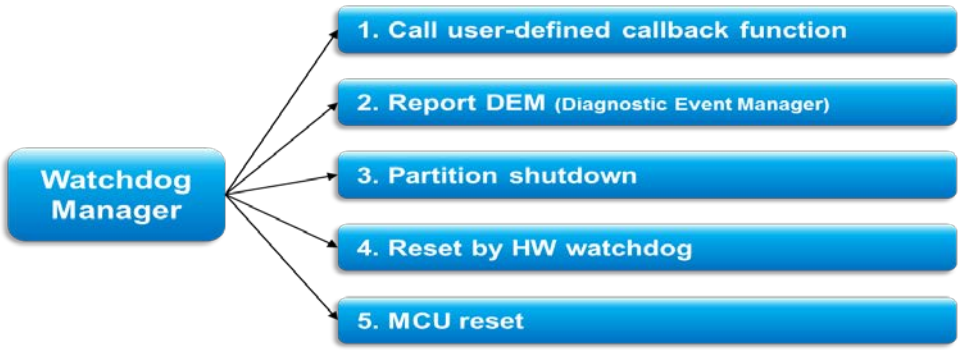


그림 10. WdgM이 수행할 수 있는 오류 처리 작업

이제까지 설명한 program flow monitoring이 제공하는 세 가지 기능이 WdgM에 의해서 어떻게 수행이 되는지 살펴보도록 하겠다. Program flow monitoring은 크게 두 부분에서 이루어 진다. 첫째, program flow monitoring의 대상이 되는 SE, 둘째, OS scheduler에 의해 주기적으로 호출되는 WdgM의 main function. 전체 program flow monitoring의 과정을 그림 11를 통해 설명한다. 먼저 SE 문맥에서 SE가 수행 중에 체크포인트를 만나게 되면 실제로는 이 체크포인트에 해당하는 지점에 WdgM_CheckpointReached라는 함수를 호출하도록 되어있다. 이 함수는 먼저 이 함수를 호출한 체크포인트가 alive supervision, deadline supervision, logical supervision 중 어떤 것들을 사용하는지 체크하고, 만약 alive supervision을 사용한다면, alive supervision을 나타내는 변수인 alive indication counter를 증가시킨다. 그리고, deadline supervision과 logical supervision을 사용하는 체크포인트라면 deadline supervision과 logical supervision을 수행하게 되고 그 결과 값인 상태 정보를 업데이트한다. 이렇게 WdgM_CheckpointReached 함수에 의해 갱신된 각 supervision의 정보를 OS scheduler에 의해 주기적으로 불려지는 WdgM_MainFunction함수가 이용하게 된다. 이 함수는 SE 단위로 SE가 alive supervision을 사용한다면, 각 alive supervision을 수행하여 결과를 업데이트하고, WdgM_CheckpointReached함수에 의해 결정된 해당 SE의 deadline supervision과 logical supervision의 결과를 종합하여 SE의 상태 정보를 갱신한다. 각 SE의 상태 정보가 갱신되고 나면 이 것을 가지고 global supervision status 를 갱신하고

WdgM은 global supervision status를 바탕으로 전체 시스템에 오류가 발생했는지 안 했는지를 확인한다.

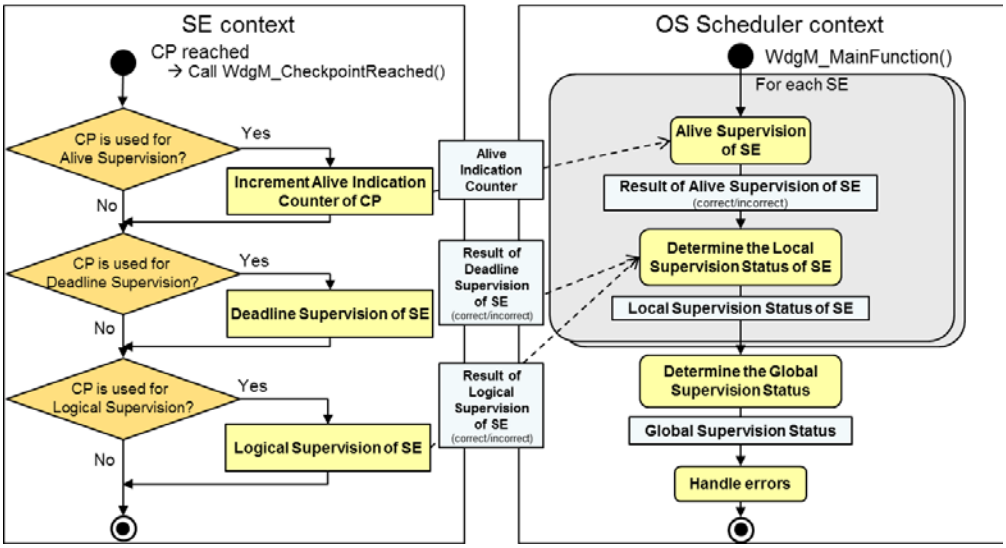


그림 11. Program Flow Monitoring의 알고리즘

이 논문에서 한계점으로 발견한 부분은 program flow monitoring에서 deadline supervision이므로, deadline supervision에 대한 상세한 설명을 추가하도록 한다. 앞서 그림 11에서 SE 문맥에서 특정 체크포인트를 지나게 되면 WdgM_CheckpointReached함수를 호출하게 되고 이 함수는 해당 체크포인트가 어떤 supervision에 연관되었는지 확인하여 해당 supervision을 수행한다. 만약 deadline supervision에 연관된 체크포인트라면 그림 12에서처럼 해당 체크포인트가 deadline supervision에서 사용되는 체크포인트 중에서 첫 번째(DSC)에 해당하는지 아니면 두 번째(DEC)에 해당하는지 확인한다.

만약, DSC와 DEC에 해당 되지 않는다면 그대로 수행 종료 후, logical supervision에 해당하는 체크포인트 인지 확인한다. 만약 그렇지 않다면, DSC 인지 DEC인지 확인하고, DSC인 경우, 현재 시스템의 timestamp를 확인하여 DSC timestamp에 기록하게 된다. 만약 DEC인 경우, DSC timestamp가 기록 되었는지 확인한 뒤, 기록되어 있지 않다면 그대로 수행을 종료하고, 그렇지 않다면 현재 timestamp와 DSC timestamp와의 시간 차를 측정하고 이 측정값이 미리 정의된 maximum 값과 minimum 값 내에 존재하면 deadline supervision의 결과를correct로, 그렇지 않다면 deadline supervision의 결과를 incorrect로 결정한다[7].

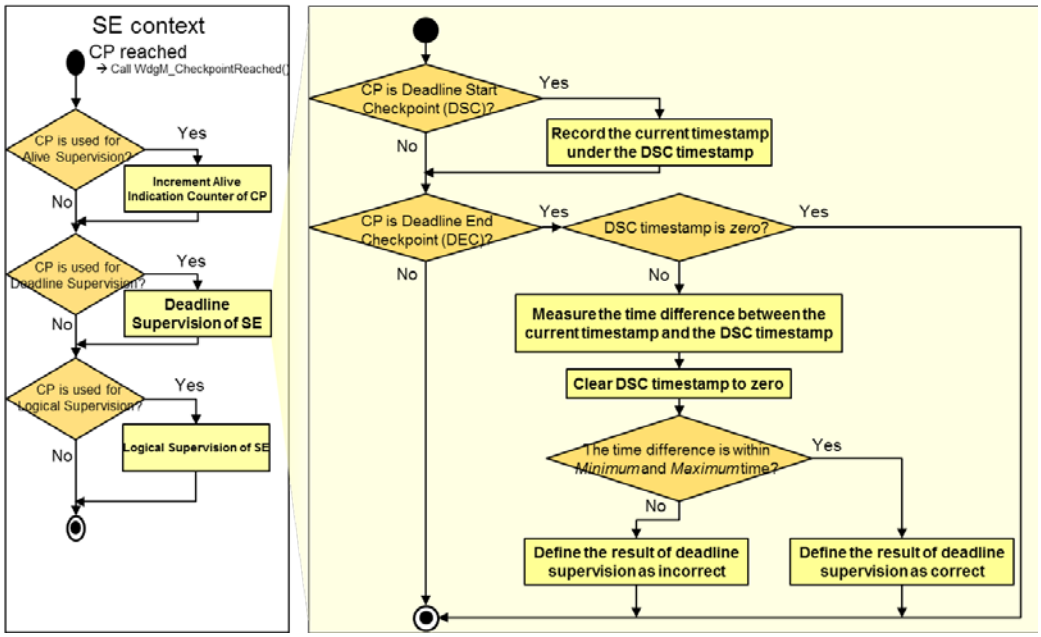


그림 12. Deadline Supervision의 알고리즘

■ E2E 보호

E2E 보호의 기능은 소프트웨어 컴포넌트 간 통신에 사용되는 데이터가 안전 관련일 경우 전송되는 데이터가 통신 연결 상에서 생길 수 있는 fault를 방지하여 다른 소프트웨어 컴포넌트로 fault가 전가되지 않도록 한다. E2E 보호 기능의 사용 주체는 데이터를 전송하는 소프트웨어 컴포넌트(Sender)와 데이터를 수신하는 소프트웨어 컴포넌트(Receiver)가 된다. E2E 보호 기능은 라이브러리 형태로 구현되어 있으며, E2E 라이브러리(E2E Library)에 그 기능이 기술되어 있다. E2E 보호를 사용하고자 하는 소프트웨어 컴포넌트는 전송하는 데이터에 E2E 헤더가 붙게 된다. E2E 보호의 기능은 4 가지 메커니즘을 아래와 같이 사용한다.

- 카운터;
- 데이터 ID;
- 순환 덧붙임 검사(Cyclic Redundancy Check, CRC);
- 타임아웃 탐지(Timeout Detection).

카운터 메커니즘은 sender가 데이터 송신 시, 각 데이터마다 E2E 헤더에 존재하는 카운터 값을 증가시켜 receiver에서 각 데이터의 E2E 헤더에 존재하는 카운터를 확인하여 통신 관련 fault 중 데이터의 반복, 소실, 추가, 부정확함, 순서, 일시 전송 중단으로 인한 fault를 검출하는 기능을 제공한다. 데이터 ID 메커니즘은 소프트웨어 컴포넌트가 사용하는 포트마다 유일한 ID를 부여하여 통신 관련 fault 중 receiver가 sender가 아닌 다른 소프트웨어 컴포넌트로부터 데이터를 받아 발생하는 fault를 방지하는 기능을 제공한다. 순환 덧붙임 검사는

통신 관련 fault 중 통신 중에 데이터가 오염될 경우 이를 확인할 수 있게 하여 fault가 전가되지 않는 기능을 제공한다. 마지막으로, 타임아웃 탐지는 receiver에서 이전에 수신된 데이터를 기준으로 현재 수신된 데이터가 얼마나 오래 되었는지 측정하여 그 데이터의 신선도(Freshness)에 따라 데이터를 사용할지 폐기할지 결정한다. 이 기능을 통해 통신 관련 fault 중 데이터의 소실, 지연, 일시 전송 중단으로 인한 fault를 검출할 수 있다. E2E 보호 기능은 선택적으로 사용할 수 있으며, AUTOSAR에서는 각 기능을 조합한 프로필을 제공한다.

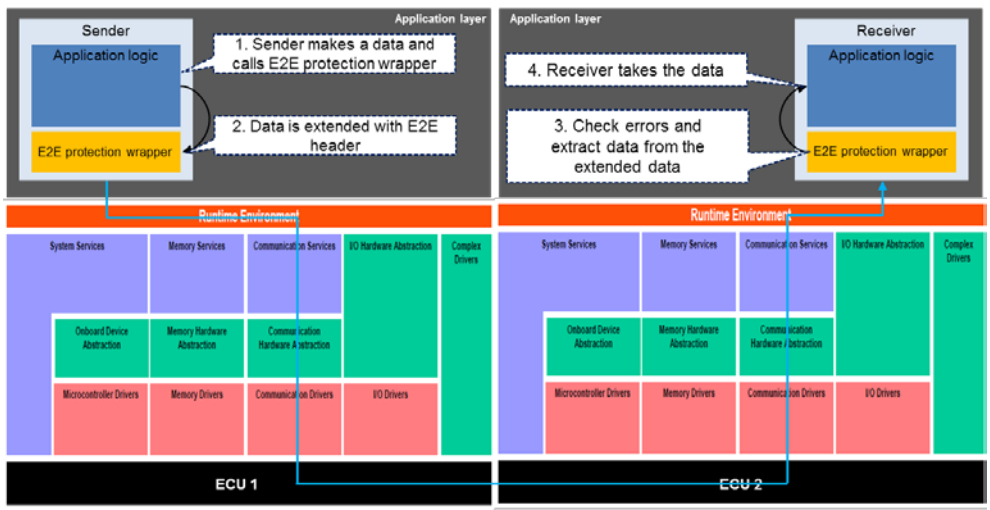


그림 13. E2E 보호 기능 메커니즘

E2E 보호의 기능은 그림 13과 같다. Sender는 수행 중에 receiver에게 데이터를 전송하고자 한다. 이때 sender는 E2E 라이브러리에 정의된 E2E protection wrapper를 호출하고 E2E

protection wrapper는 sender가 전송할 데이터에 앞서 설명한 E2E 헤더를 붙이고 RTE 호출을 통해 데이터를 전송한다. 하위 계층을 거쳐 receiver가 있는 ECU에 도달한 데이터는 마찬가지로 통신 스택을 거쳐 RTE를 통해 receiver에 있는 E2E protection wrapper에 도착하게 되고, 이 E2E protection wrapper는 전송된 데이터의 E2E 헤더를 E2E 보호기능을 기반으로 분석한 뒤, receiver에게 데이터를 전달한다.

- 메모리 분할

메모리 분할 기법은 소프트웨어 컴포넌트 간 간섭을 방지한다. 즉, 특정 소프트웨어 컴포넌트에서 발생한 메모리 관련 fault가 발생할 경우, 다른 소프트웨어 컴포넌트로 이 fault가 전가되지 않도록 한다. 메모리 분할 단위는 OS-Application이며, OS-Application은 소프트웨어 컴포넌트 집합과 AUTOSAR 운영체제의 객체들인 인터럽트 서비스 루틴 집합, 알람 집합, 등을 포함한다. 메모리 분할 기법은 OS-Application 단위로 메모리 관련 fault 전가를 방지하며, 만약 이러한 fault가 발생할 경우, AUTOSAR 운영체제는 해당 OS-Application을 재 시작하거나 AUTOSAR 운영체제 전체를 재 시작하는 오류 처리 작업을 할 수 있다. 그림 14에서처럼 SWC 1과 SWC 2는 같은 OS-Application에 속하고, SWC 1과 SWC 2는 일부 메모리 영역을 공유한다고 가정하자. SWC 1에 잘못된 메모리 접근으로 인해 SWC 2가 접근하는 메모리 영역이 변경되는 fault가 발생한다. 따라서 SWC 1으로 인한 fault가 SWC 2에게 전가 되지만, 다른 OS-Application에 속한 SWC 3와 SWC 4는 아무런 영향을 받지 않는다.

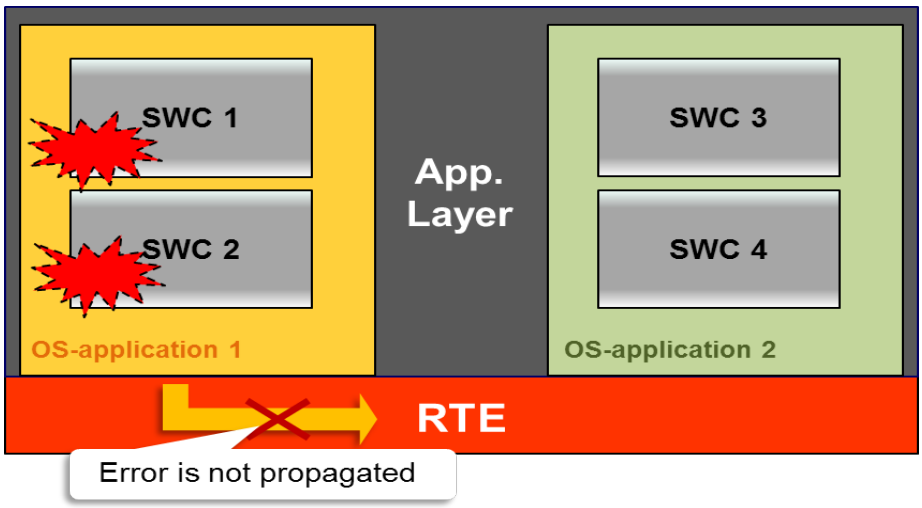


그림 14. 메모리 분할 메커니즘 예시

- 통신 스택 관련 기능

AUTOSAR BSW의 서비스 계층에 속하는 COM 모듈이 담당하는 기능으로, 통신 스택 상에서 발생 가능한 fault를 탐지하고 방지한다. 세부적으로 두 가지 기능을 제공하며, 첫째는 데이터 순서 제어(Data Sequence Control), 둘째는 통신 보호(Communication Protection)이 있다. 두 기능에 대한 상세 설명에 앞서 COM 모듈에서 다루는 데이터의 단위에 대해서 설명이 필요하다. COM 모듈은 소프트웨어 컴포넌트로부터 ECU 외부 통신 시, 사용된다. 일반적으로 소프트웨어 컴포넌트가 전송하는 데이터의 크기는 문자형, 정수형과 같은 기본 데이터 형이므로 이러한 데이터를 하나하나 관리하기 매우 비효율적이다. 따라서 소프트웨어 컴포넌트들이 ECU 외부 통신을 위해 COM 모듈에 보낸 데이터를 그룹화하여 다루게 된다. COM 모듈이 다루는 데이터

단위를 프로토콜 데이터 단위(Protocol Data Unit, PDU)라 한다. PDU는 헤더(Header)와 페이로드(Payload)로 구성된다.

데이터 순서 제어의 기능은 PDU의 헤더에 포함된 PDU 카운터를 이용하여 PDU들의 순서를 확인하고 필요 없는 PDU는 과감히 제거한다. 송신 측에서 데이터 순서 제어 기능 이용 시, 최초 PDU 카운터는 0이며 PDU를 전송할 때마다 하나씩 증가한다. 수신 측에서 PDU를 수신하면 카운터를 확인하여 카운터 순서와 관계없는 PDU는 제거 후에 PDU를 구성하는 데이터들을 각자 맞는 소프트웨어 컴포넌트에게 전달한다.

통신 보호 기능은 안전과 관련된 PDU를 복제하여 소실과 오염을 방지한다. 송신 측에서 전송하고자 하는 PDU를 두 개 복제하여 동일한 내용을 가지는 총 세 개의 PDU를 전송하게 된다. 수신 측에서 세 PDU를 서로 비교와 투표를 통해서 올바르게 전송된 PDU를 선택하게 된다[9].

- 시간 동기화

소프트웨어 컴포넌트와 BSW 모듈에게 동기화된 시간을 제공함으로써 분산된 소프트웨어 컴포넌트들과 BSW 모듈들의 시간 차이로 인한 시간 관련 fault를 사전에 방지하는 기능을 제공한다. 이 기능은 BSW 서비스 계층의 동기화된 시간 기준 매니저(Synchronized Time Base Manager, StbM)가 담당한다. 동기화된 시간을 필요로 하는 소프트웨어 컴포넌트와 BSW 모듈을 고객이라 하며, StbM은 고객에게 동기화된 시간을 제공하기 전에 분산된 StbM 간의 시간을 동기화해야

한다. 이를 위해 NTP와 같은 시간 프로토콜이 사용된다. StbM 간의 시간 동기화가 완료되면 고객이 할 때, 동기화된 시간 정보를 전송하게 된다[10].

# 제 3 장 문제 정의

이 장에서는 본 논문에서는 앞서 2장에서 설명한 AUTOSAR의 safety mechanism이 지닌 한계점을 설명한다. 이를 위해 먼저 본 논문이 설명하고 자하는 대상 시스템을 설명하고 AUTOSAR의 safety mechanism의 두 가지 한계점에 대해 설명한다. (1) 프로그램 흐름 모니터링, (2) E2E 보호.

## 3.1 시스템 모델

본 논문에서 설명할 문제와 개선점의 대상이 되는 시스템은 AUTOSAR 소프트웨어 아키텍처이며 2장에서 상세히 설명하였다. 그림 15와 같이 AUTOASR의 safety mechanism은 E2E 라이브러리를 제외하면 모두 BSW에 구현되어 있으며, 이들은 하나의 모듈 혹은 모듈 안의 한 기능 형태로 존재한다.

## 3.2 프로그램 흐름 모니터링의 한계점

프로그램 흐름 모니터링은 ISO 26262에서 정의한 소프트웨어 상 발생 가능한 fault들 중, blocking of execution fault를 완벽히 다루지 못한다. 구체적으로, 프로그램 흐름 모니터링의 세 가지 기능 중

deadline supervision에 한계점이 보인다. Deadline supervision을 사용하는 SE가 CheckPointStart를 지나면 WdgM_CheckpointReached 함수를 호출한다. 이 함수는 SE의 ID와 체크 포인트 ID를 인자로 받으며, 이 함수가 불리어진 시각을 AUTOSAR 운영체제로부터 틱(Tick)값을 받아 CPU의 클락 속도를 고려하여 시간으로 변환한 뒤, WdgM에 존재하는 이 함수를 호출한 SE의 체크 포인트에 해당하는 시간 변수에 저장한다.

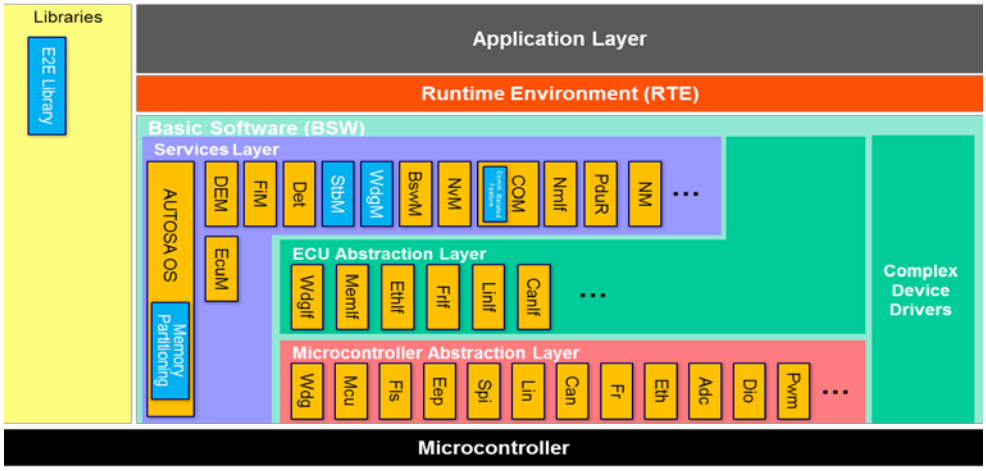


그림 15. AUTOSAR 소프트웨어 아키텍처 상 존재하는 Safety Mechanism [2]

이후, SE가 CheckPointEnd를 지나면 마찬가지로 WdgM_CheckpointReached 함수를 호출하고 앞에 설명한 동일한 작업을 거쳐 이 함수가 호출된 시각을 구한다. 추가적으로 앞서 측정한 CheckPointStart에서의 WdgM_CheckpointReached 함수를 호출한 시각과 현재 시각의 차를 계산하여 이 시간이 WdgM 내 정의된 해당 SE의 최소 수행 시간과 최대 수행 시간 내에 존재하는지 확인한다.

이러한 메커니즘은 한 가지 문제점이 존재한다. 즉, 만약 deadline supervision을 사용하는 SE가 CheckPointStart를 지난 뒤, 특정 원인에 의하여 수행이 중단 혹은 지연될 경우, SE가 CheckPointEnd를 지날 때 비로소 SE가 미리 정의한 최대 수행 시간을 벗어남을 탐지하고 오류 처리 작업을 한다. 이것은 차량에서 매우 심각한 상황을 야기할 수 있다. 예를 들어, 그림 16처럼 YAW&G 센서가 부착된 에어백 ECU가 센서 정보를 처리한다. 하지만 이 센서 정보는 차량자세제어시스템(Electronic Stability Control, ESC)에서도 필요하기 때문에 ESC ECU는 CAN 통신을 통해 에어백 ECU로부터 센서 정보를 받게 된다. ESC ECU 내에는 에어백 ECU로부터 YAW&G 센서 정보를 받는 receiver 소프트웨어 컴포넌트(SWC), receiver SWC로부터 YAW&G 센서 정보를 전달받아 4 개의 브레이크의 제어를 결정하는 control logic SWC, control logic SWC로부터 제어 정보를 받아 실제 브레이크를 제어하는 actuator control SWC가 있다.

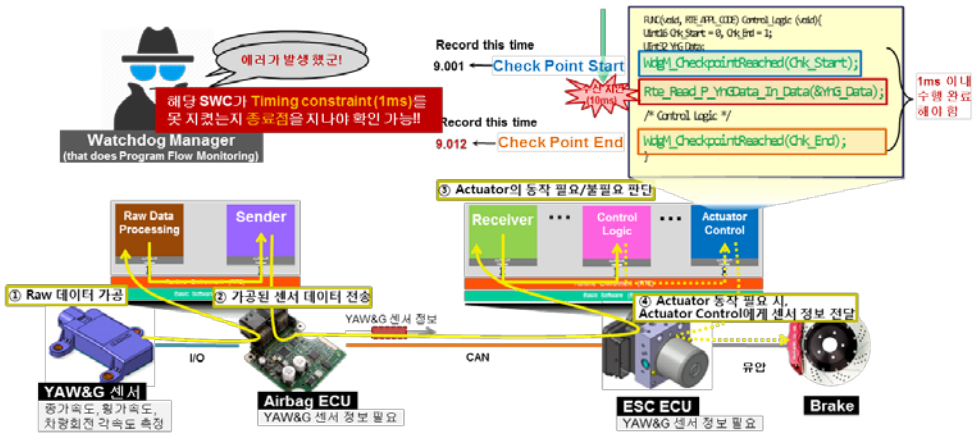


그림 16. 현재 AUTOSAR의 프로그램 흐름 모니터링의 문제 상황 예시

이때, Control SWC는 안전과 직결되기 때문에 브레이크 제어를 결정하는 코드 구간을 deadline supervision을 이용해 확인한다.

코드 구간이 1ms 내에 수행이 완료되어야 하지만, CAN 통신에 과부하가 발생하여 YAW&G 센서 정보의 수신이 지연되면 receiver SWC가 YAW&G 센서 정보를 제때 control logic SWC에게 전달할 수 없고 이로인 해 control logic SWC는 CheckPointStart를 지난 뒤, 지연될 수 밖에 없다. 만약, Receiver로부터 10ms 뒤에 YAW&G 센서 정보를 받게 되면, 10ms 뒤에 CheckPointEnd를 지나게 되며, 이 시점이 되어서야 WdgM은 오류가 발생했음을 확인하고 오류 처리 작업을 수행하게 된다.

하지만 이렇게 늦게 전달된 센서 정보는 현재 상태의 차량을 정확히 나타내지 못하고 이 센서 정보를 이용하여 차량의 상태를 결정하는 것은 결국 차량에 심각한 위험을 초래한다.

### 3.3 E2E 보호의 한계점

E2E 보호의 한계점은 바로 ISO 26262에서 정의한 소프트웨어 상 발생 가능한 fault 중 exchange of information 분류에 해당하는 delay of information fault를 전혀 고려하지 않고 있다는 것이다. 구체적으로, 현재까지 개발된 AUTOSAR는 E2E 보호 기능으로 4 가지를 제공한다. (1) 카운터, (2) 데이터 ID, (3) 순환 덧붙임 검사, (4) 타임아웃 탐지. 이 중에서 마지막 기능인 타임아웃 탐지는 receiver에서 현재 도착한 데이터의 도착 시각이 바로 이전에 도착한 데이터의 도착 시각과

비교하여 미리 정의한 데드라인(deadline)을 초과하는지 확인하는 기능이다. 하지만 AUTOSAR에서 이 기능을 담당하는 API를 제공하지 않고 카운터를 활용하여 타임아웃 탐지를 구현하도록 가이드라인만 제시하고 소프트웨어 컴포넌트 개발자가 직접 구현해야 한다. 하지만 이는 문제가 될 수 있다. 즉, 서로 다른 부품업체가 각자가 개발한 ECU의 SWC에 타임아웃 기능을 사용하고자 독자적으로 API를 개발하게 되면, 동일한 타임아웃 기능을 수행하는 다양한 API들이 혼재하며 결국 이를 통합해야 하는 자동차 제조사 입장에서 차량 전체 시스템을 구성하는 과정에서 문제가 발생할 수 있다. 이러한 문제 상황은 그림 17에 잘 묘사되어 있다.

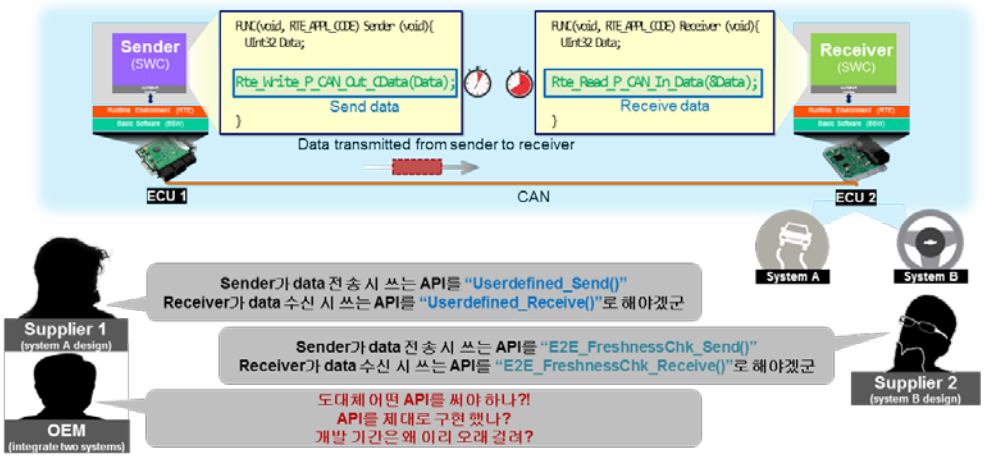


그림 17. 현재 AUTOSAR의 E2E 보호의 문제 상황 예시

# 제 4 장 개선된 AUTOSAR의 Safety Mechanism

이 장에서는 3장에서 설명한 AUTOSAR safety mechanism이 지닌 문제점을 해결하는 개선안을 설명한다. 먼저, 프로그램 흐름 모니터링의 개선안을 설명하고, 다음으로 E2E 보호에 대한 개선안을 설명한다.

## 4.1 개선된 프로그램 흐름 모니터링

버전 4.2의 프로그램 흐름 모니터링에서 deadline supervision의 문제를 해결하기 위해 SE가 체크 포인트를 지나는 시각을 기록하는 곳을 변경한다. 즉, SE가 체크 포인트를 지날 때, WdgM_CheckpointReached 함수를 호출하고 이 함수가 SE가 현재 통과한 체크 포인트의 시각을 측정하고 만약 두 번째 체크 포인트를 지나 이 함수를 호출했다면, 두 체크 포인트 사이의 시간 간격을 계산하는 역할을 한다. 즉, 오류를 판단하기 위해서는 반드시 두 번째 체크 포인트를 지나 WdgM_CheckpointReached 함수를 호출 해야한다. 따라서 본 논문에서는 첫 번째 체크 포인트를 지나 WdgM_CheckpointReached 함수로부터 체크 포인트 통과 시각을 WdgM이 받으면, WdgM은 주기적으로 현재 시각을 AUTOSAR OS로부터 틱 값을 받아 시각으로 변환한 뒤, CheckPointStart 통과 시각과 차를 구하여 미리 정의한 최대 수행 시간과 비교한다. 이를 통해

WdgM이 deadline supervision을 사용하는 SE에 문제가 발생한 상황을 보다 빠르게 인지하여 오류 처리 작업을 보다 빠르게 호출할 수 있도록 한다.

### 4.2 개선된 E2E 보호

현존하는 최신 버전의 AUTOSAR가 타임아웃 탐지 기능에 대한 표준 API가 존재하지 않으므로 이에 대한 표준 API를 설계하여 제안한다. 표준 API의 명칭은 E2E_FreshnessCheck 이며, 기존 E2E 라이브러리에 사용된 API들과 일관성을 유지하기 위에 앞머리에 E2E를 붙였다. 이 API를 사용하기 위해선 먼저 바로 이전에 도착한 데이터의 시각을 저장하는 변수가 필요하다. E2E_FreshnessCheck API는 먼저 호출 즉시, AUTOSAR 운영체제로부터 틱 값을 받아온다. 이 틱 값을 CPU 클락 수를 고려하여 시간으로 변환한다. 만약 이 API가 최초로 도착한 데이터에 대한 시각을 계산한 경우라면 이 시각만 저장한 뒤 함수를 종료한다. 두 번째 도착한 데이터를 대상으로 앞서 한 작업을 수행한 뒤, 이전에 도착한 데이터의 시각과 비교한다. 이후, 미리 정의해둔 데드라인 시각과 비교하여 데드라인 시각보다 적은 시간이 걸렸다면 정상 상태임을 나타내고 그렇지 않다면 비정상 상태임을 리턴한다.

# 제 5 장 실험 및 검증

이 장에서는 앞서 제안한 개선된 AUTOSAR의 safety mechanism이 3장에서 정의한 현재 버전의 AUTOSAR safety mechanism의 한계점을 보완하는지 실험환경을 구축하고 검증한다.

## 5.1 실험 환경

개선된 AUTOSAR의 safety mechanism을 구현하고 검증하기 위해 아래 표 1과 같이 하드웨어와 소프트웨어 실험 환경을 구성하였다. 주목할 점은 EB의 tresos가 버전 3.1의 AUTOSAR 표준을 준수한다.

| Hardware | | | Software | |
|---|---|---|---|---|
| TriCore™ Starter Kit (2 EA) | MCU | TC1797 (TriCoreTM V1.3.1 CPU 180MHz) | AUTOAR version | Release 3.1 |
| | Memory | 4MB flash memory | SWC modeling | SystemDesk 3.0 |
| | I/O | 219 general purpose I/Os | ECU Configuration | EBtresos 10.0 |
| | Comm. | 2 CAN transceivers<br>2 FlexRay transceivers | Downloader | TASKING VX-Toolset for TriCore V3.4r1 |

표 1. 실험 환경에 사용된 하드웨어와 소프트웨어 상세

본 논문에서 개선한 것은 safety mechanism 중 program flow monitoring과 E2E 보호이다. 2장에서 언급한 것과 같이 AUTOSAR

버전 3.1에서는 safety mechanism이 정립되어 있지 않다. 다만 program flow monitoring을 담당하는 WdgM는 존재하고 있으며 alive supervision 기능만 지원한다. 따라서, 본 논문에서 제안한 개선안을 구현하기 위해 그림 18처럼 AUTOSAR 버전 3.1에 개선하고자 하는 기능을 담당하는 WdgM 모듈에 deadline supervision 기능을 추가하고, E2E 라이브러리를 적용한다. 따라서 AUTOSAR 버전 3.1에서 본 논문에서 검증하고자 하는 program flow monitoring의 deadline supervision과 E2E 보호의 freshness API는 버전 4.2와 같은 기능을 제공한다. 이렇게 구축된 실험 환경에서 제안한 개선된 safety mechanism을 구현하였다.

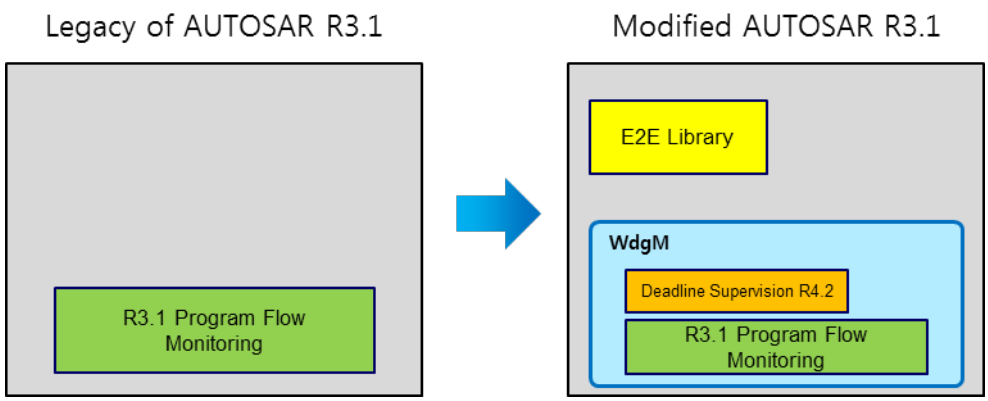


그림 18. 기존 AUTOSAR 버전 3.1에 버전 4.2의 프로그램 흐름 모니터링과 E2E 보호가 추가된 수정된 AUTOSAR 버전 3.1

## 5.2 실험 구성

본 논문에서는 AUTOSAR safety mechanism의 두 가지를 개선한다. 각각의 개선안에 대한 실험 구성은 다음과 같다. 먼저, program flow monitoring에 대한 실험 구성은 1개의 하드웨어 보드를 사용하며, 소프트웨어는 앞서 설명한 수정된 AUTOSAR 버전 3.1이 탑재된다. 실험 시나리오는 다음과 같다. Deadline supervision을 사용하는 SE가 존재한고 이 SE는 CheckPointStart를 지난 뒤, CheckPointEnd를 지나기 전, 임의의 시간동안 동작을 멈춘다. 그리고 개선된 E2E 보호를 검증하기 위해 2개의 하드웨어 보드를 사용하며, 하나의 보드에는 데이터를 전송하는 소프트웨어 컴포넌트가, 다른 하나의 보드에는 데이터를 수신하는 소프트웨어 컴포넌트가 탑재된다. 각 보드에는 앞 실험과 마찬가지로 수정된 AUTOSAR 버전 3.1이 탑재된다. 실험 시나리오는 다음과 같다.

Sender가 E2E 보호를 이용하여 전송하고자 하는 데이터에 E2E 헤더를 추가한다. 이후, 데이터를 일정 시간 간격으로 전송한다. receiver는 E2E 보호를 이용하여 수신된 데이터의 페이 로드를 추출하며, freshness API를 이용하여 이전에 도착한 데이터의 시각과 현재 도착한 데이터의 시각을 비교하여 미리 정의한 시간 간격 내에 도착하였는지 확인한다. 이때, sender는 임의의 시점에서 임의의 시간 간격 동안 데이터를 송신하지 않는다.

### 5.3 실험 평가

개선된 program flow monitoring이 구현된 WdgM은 SE가 CheckPointStart를 통과하여 CheckPointEnd에 도달하기 전 일시적으로 멈출 때, CheckPointStart를 지난 시점부터 일정 시간마다 현재 시각과 CheckPointStart 통과 시각의 차를 계산하여 미리 정의된 최대 수행 시간과 비교, 확인한다. SE가 수행이 중단 되고 미리 정의된 최대 수행 시간을 넘었을 때, WdgM은 오류 처리 작업을 수행하였다. 이를 통해 개선된 program flow monitoring이 정상 작동함을 검증하였다.

개선된 E2E 보호 기능이 추가된 E2E 라이브러리를 사용하는 receiver는 데이터가 도착할 때마다 freshness API를 이용하여 현재 도착한 데이터 바로 이전에 도착한 데이터의 시각과 현재 도착한 데이터의 시각을 비교하여 미리 정의된 시간 간격 내에 존재하는지 확인한다. Sender의 데이터 전송을 임의의 시점에서 정상 전송 시간 간격이상으로 지연시켰을 때, receiver는 정상 전송 시간 간격이 지난 후, 오류 처리 작업을 수행하였다. 이를 통해 개선된 E2E 보호 기능이 정상 작동함을 검증하였다.

# 제 6 장 결 론

2003년부터 개발이 시작된 AUTOSAR는 차량용 전기/전자 시스템에 대한 기능 안전 표준인 ISO 26262가 제정된 이후, 기능 안전 요구사항을 만족시키기 위하여 safety mechanism을 도입한다. 하지만 현재 존재하는 버전 4.2 AUTOSAR의 safety mechanism은 프로그램 흐름 모니터링과 E2E 보호에서 문제점이 보인다.

본 논문에서는 이 두 문제점을 설명하고 해결할 수 있는 방안을 제안하였다. 해결책은 각각 AUTOSAR 소프트웨어에 구현되어 하드웨어 보드에서 검증이 완료되었다. 본 논문에서 제안한 기법을 통해 AUTOSAR의 소프트웨어 신뢰성 향상에 기여할 수 있었다.

# 참고 문헌

[1] Jason Dorrier, “BMW Forecasts Cars Will Be Highly Automated by 2020, Driverless by 2025,” SingularityHUB, 2013.

[2] AUTOSAR, “Layered Software Architecture,” http://www.autosar.org, 2015.

[3] Fürst, Simon, et al. "AUTOSAR–A Worldwide Standard is on the Road." 14th International VDI Congress Electronic Systems for Vehicles, Baden-Baden. Vol. 62. 2009.

[4] AUTOSAR, Specifications, http://www.autosar.org.

[5] ISO, ISO 26262 Road vehicles –Functional safety–, http://www.iso.org

[6] AUTOSAR, “Technical Safety Concept Status Concept Report,” http://www.autosar.org, 2015

[7] AUTOSAR, “Specification of Watchdog Manager,” http://www.autosar.org, 2015.

[8] AUTOSAR, “Specification of SW-C End-to-End Communication Protection Library,” http://www.autosar.org, 2015.

[9] AUTOSAR, “Specification of Operating System,” http://www.autosar.org, 2015.

[10] AUTOSAR, “Specification of Communication,” http://www.autosar.org, 2015.

[11] AUTOSAR, “Specification of Synchronized Time-Base Manager,” http://www.autosar.org, 2015.

## Abstract

# Improved Safety Mechanisms of AUTOSAR for Enhancing Software Reliability

Hyunjun Yoon
Electrical and Computer Engineering
The Graduate School
Seoul National University

Many automotive companies are trying to develop the autonomous vehicle. Recently, major vehicle OEMs adopt ADAS in their luxury models. It is needed for ADAS to install many sensors such as Radar, LiDAR, and Camera. Therefore, the size of automotive software becomes large and complex. To deal with such problem properly, automotive companies in Europe made AUTomotive Open System Architecture (AUTOSAR). Moreover, ISO 26262, the functional safety for automotive equipment standard, appeared to prevent the malfunctioning of automotive electric/electronic systems. AUTOSAR adopted safety mechanisms to satisfy the functional safety requirements of ISO 26262. However, the safety mechanisms that exist in latest AUTOSAR version (R 4.2) cannot cover all faults defined in ISO 26262. First, if needed, program flow monitoring which checks the correct

execution of software cannot detect an error which occurs in specific execution situations. In addition, E2E protection which detects communication related faults in data exchange between application software does not provide the standard API but the guideline for using timeout detection. Therefore, this paper proposes the improved safety mechanism of AUTOSAR for covering all faults defined in ISO 26262. First of all, this paper proposes the improved program flow monitoring which can detect correct execution of software in all execution situation. Furthermore, this paper proposes the standard API which is used to check the freshness of exchanged data using E2E protection. Finally, this paper proves that the proposed AUTOSAR safety mechanisms can satisfy the above problems with experiment.

**Keywords:** AUTOSAR, Automotive Software Platform, Safety Mechanism, Program Flow Monitoring, E2E Protection
**Student Number:** 2014-21633