Research Report

연구보고서 2018-011

소프트웨어안전 확보를 위한 개발프로세스 적용 확산 방안 연구

Study on Activation of Development Process for Software Safety Assurance

진회승/송지환/권영환/이낙선/양회석/장은미/김문희

2018.12

SPRi 소프트웨어정책연구소

이 보고서는 2018년도 과학기술정보통신부 정보통신·방송 연구 개발사업에서 지원받아 제작한 것으로 과학기술정보통신부의 공식입장과 다를 수 있습니다.

목 차

표 목 차

그 림 목 차

요 약 문

1. 제 목

소프트웨어 안전 확보를 위한 개발 프로세스 적용 확산 방안 연구

2. 연구 목적 및 필요성

산업별로 소프트웨어 의존성과 복잡성이 증가함에 따라 소프트웨어의 결함으로 인한 사고 발생 가능성이 높아지게 되었으며, 소프트웨어로 인한 사고 발생 시 사회·경제적 피해가 막대할 가능성이 있으며, 소프트웨어 안전이 국민 안전 확보를 위한 핵심요소로 부각되고 있다. 그 동안 크게 부각되지 않던 소프트웨어안전이 중요해짐에 따라 그를 구현할 수 있는 방법에 대한 연구가 필요하게 되었다.

소프트웨어안전 선진국은 시스템안전 표준과 함께 소프트웨어안전 표준을 정하고, 소프트웨어안전을 지키기 위한 활동을 하고 있다. 소프트웨어안전 표준의 주요 활동은 안전 프로세스 적용이라 할 수 있다. 하드웨어의 안전은 안전 무결성 수준(Safety Integrity Level)을 결정하고 그에 맞는 실제 고장률을 측정하여 그 달성도를 확인한다. 소프트웨어안전의 경우는 실재 고장률 측정이 불가능하기 때문에 소프트웨어안전을 위한 개발 프로세스를 정하고 이에 따라 개발한 경우 소프트웨어안전이 구현되는 것을 전제로 하고 있다.

2015년과 2016년에 소프트웨어정책연구소에서 수행한 『소프트웨어안전 산업 동향 조사』에서는 국내의 경우 소프트웨어안전 프로세스 활동 중에서도 가장 중요한 위험도 분석이나 안전 메카니즘 분석, 설계보다는 테스트위주의 활동을 위주로 활동한다고 조사되어, 소프트웨어안전 확보의 주요 활동인 소프트웨어안전 프로세스 적용에 대한 적용현황을 조사하고, 활성화하는 정책의 필요성이 제기되었다.

본 연구의 목적은 소프트웨어안전 확보를 위한 소프트웨어 개발 프로세스 적용 현황을 면밀히 확인하고, 제품 및 기업 현황에 맞는 소프트웨어 안전 개발 프로세스의 적용과 확산 방안의 도출이다.

3. 연구의 구성 및 범위

본 연구는 총 6장으로 구성되어 있다.

제 1장에는 서론으로 소프트웨어 안전 확보를 위한 개발 프로세스 적용 확산 방안 연구의 배경과 필요성, 목적, 내용 및 본 연구를 수행한 연구 수행 방법을 제시한다.

제 2장에는 소프트웨어 안전 확보를 위한 요소들을 확인하고, 소프트웨어 개발 프로세스, 소프트웨어 안전 프로세스에 대해 정리한다. 본 장의 통해 정리된 프로세스를 기반으로 안전 개발 프로세스 조사지를 구성한다. 또한 문제점 도출 및 개선 방안 마련을 위해 소프트웨어 안전 미적용 사고 사례 및 표준 적용 사례를 분석한다.

제 3장에는 소프트웨어 안전 개발 프로세스 실태 조사를 위한 조사 내용 및 조사 대상에 대한 내용으로 비표준 분야는 IEC 61508, 표준분야에서 자동차 분야는 ISO 26262, 철도분야는 IEC 62279, 항공 분야는 DO-178C를 중심으로 소프트웨어 안전 개발 프로세스 및 단계별 세부 활동, 안전 기법 및 산출물을 조사하여 실태 조사지를 구성한다.

조사 대상은 소프트웨어정책연구소, 정보통신산업진흥원, 각 분야별 유관 협회 등에서 조사 대상 목록을 수집하고, 유효대상 선별 작업을 수행 한 후 사전 전화로 실태 조사 대상을 확정하고 개별 방문을 통한 조사지 조사 및 심층 인터뷰를 진행한다.

제 4장에는 조사지 조사 및 심층 인터뷰 결과를 정리하고 안전 프로세스 적용 현황 조사 결과를 분석하였다. 조사지 응답자 특성 현황, 조사 결과에 대한 객관적 분석을 시행하고, 공통질문과 세부질문의 관계, 이전 질문과 다음 질문의 관계 등의 분석하여 결과에 대한 해석을 추가하였다.

제 5장은 문제점 분석 및 개선 방안에 대한 내용으로, 소프트웨어 안전 개발 실무자들이 느끼는 현장에 적용이 어려운 문제들과 이를 해결하기 위한 개선 방안을 제시한다.

제 6장은 결론으로 연구의 요약, 시사점과 향 후 연구, 연구의 한계점 등에 대해 기술한다.

4. 연구 내용 및 결과

본 연구에서 소프트웨어 안전 개발 프로세스 적용을 위한 확산 방안을 마련하기 위하여 필요한 국제 표준 소프트웨어 안전 관련 배경지식을 조사하였다. 소프트웨어 안전의 기본 개념을 조사하고, 안전 확보를 위해 기본적으로 수행하는 활동을 정의하였다. 소프트웨어 안전 확보를 위해서는 첫째 안전 기능이 구현이 필요하고, 둘째 안전 등급에 따른 안전 무

결성이 보장되어야 하며, 마지막으로 소프트웨어가 안전 개발 프로세스에 따라 개발되어야 한다. 본 연구의 중심내용은 소프트웨어 안전 개발 프로세스에 대한 것인데, 안전 개발 프로세스가 기존 소프트웨어 개발 프로세스와 다른 점은 위험분석과 안전 검증이 추가된다는 점이다. 이에 따라 안전 전문 인력과 안전 지식이 필요하고, 추가 비용이 소요된다.

본 연구에서는 안전 미적용 사고 사례와 안전 프로세스 적용 사례를 추가하였는데, 사고 사례로는 Therac 25 사고와 우버 자율주행차 사고를 분석하였다. 적용사례로는 해외 사례로 영국 61508 협회와 CASS Scherme 사례를 분석하였다. 국내는 안전보건공단의 활동에 대해 조사하였다.

조사는 자동차, 항공, 철도의 표준 분야와 비표준 분야로 나누어 시행되었다. 안전 프로세스 적용 현황 조사지 구성을 위해서 IEC 61508, ISO 26262, IEC 62279, DO-178C의 표준에서 소프트웨어 안전 프로세스를 정의하였다. 소프트웨어안전 프로세스의 내용이 방대하여 기본적인 소프트웨어 개발 프로세스는 제외하고 안전관련 활동에 집중하여 조사지를 작성하였다. 조사 대상이 이해할 수 있는 가능한 쉬운 용어로 조사지를 구성하였다. 소프트웨어 안전 전문 자문단과 논의하여 소프트웨어 안전개발 프로세스 적용을 위한 실태조사 항목과 질문을 확정하였다.

조사 대상 기업 현황, 표준 및 비표준 분야에 대한 공통 설문, 소프트웨어 안전 프로세스에 대한 각 분야별 설문, 향후 개선 및 지원 요구사항 등을 조사지로 구성하였다. 각 표준을 기반으로 한 조사항목은 조사를 위한 기본 자료로 사용하며, 각 조사자의 특성에 맞는 의견을 취합을 위한 부분도 조사지에 구성하였다.

각 분야별 설문의 경우 요구사항 분석, 아키텍처 설계, 상세 설계, 구현, 테스트의 공통 단계를 설정하고, 각 단계 수행여부에 대한 공통 질문을 구성하였으며, 각 산업별 특성을 가진 안전 개발 프로세스에 대한 내용은 하위 질문으로 추가 구성하였다. 표준분야로서 항공 분야는 DO-178C, 자동차 분야는 ISO 26262, 철도는 IEC 62279 기반으로 조사지를 설계하였다. 비표준 분야인 경우 기능 안전 모표준인 IEC 61508을 기준으로 조사지 항목을 설계하였다.

소프트웨어 안전 산업 확산을 위해 소프트웨어 안전 프로세스 적용을 위해 지원해야 하는 사항을 도출할 수 있도록 심층실태 조사를 실시하였다. 또한, 비표준 분야에서는 소프트웨어 안전 개념 및 적용 분야에 소프트웨어 안전 중요도에 대한 정도와 기존 소프트웨어 안전 프로세스 적용 시 문제점이 있는지에 대한 심층 실태 조사를 실시하였다.

심층 조사 분석을 위해 소프트웨어 안전성에 대한 업무를 수행하고 있는 기업을 대상으로

진행하였으며, 소프트웨어 안전에 대한 소프트웨어 사업 현황, 인력 역량, 연구소 역량 등을 기본적으로 파악한 후 업체를 선정하여 진행하였다. 방문 심층 실태조사를 위해 비표준 분야 산업, 자동차, 철도, 항공 등 소프트웨어 안전 프로세스를 적용하고 있는 기업 50개를 선정하여 심층 실태 조사를 실시하였다. 심층 실태 조사에 참여한 현황은 비표준 15개 기업, 자동차 11개 기업, 철도 11개 기업, 항공 15개 기업을 대상으로 방문 조사를 실시하였다.

조사 결과와 분석 내용은 방대하여 본 요약에는 포함하지 않는다. 4장 실태조사 및 분석 마지막 절에 종합 의견을 추가하여 조사 분석 내용을 요약 정리하였다. 소프트웨어 안전 일반 사항에 관한 공통 설문 부분을 종합적으로 분석하면 철도, 항공 부분은 안전에 대한 의식, 준비, 활동 등이 높은 수준을 보였으며, 자동차는 중간 수준, 비표준 분야는 조금 부족한 것으로 조사되었다. 비표준 분야는 위험 분석, 안전 관리 측면에서 인식, 지식, 및 수행이 부족한 것으로 조사되었다.

소프트웨어안전 개발 프로세스 적용 실태 조사를 통해 그동안 추정하고 있던 사실들에 대한 좀 더 객관적인 데이터를 산출할 수 있게 되었다. 안전 인식에 대한 문제가 계속적으로 제기되었고 이번 조사를 통해 비표준 분야의 안전 인식 제고의 필요성이 도출되었다. 안전 프로세스 적용에서 가장 어려운 점 중의 하나로 안전 전문 인력 부족 문제가 도출되었다.

실태 조사 결과 분석을 바탕으로 자동차, 항공, 철도 등 표준 분야에서 다음과 같은 문제점이 도출되었다.

- 소프트웨어 안전성 보증을 위한 기법 적용이 어려움
- 소프트웨어 안전성 확보를 상세 설계 과정에서 구현을 위한 충분한 설계가 이루어지지 않고 있음
- 소프트웨어 안전 개발 프로세스 각 단계별 세분화된 프로세스 정보가 미흡
- 테스트 단계의 산출물이 계획 대비 소프트웨어 안전성 요구사항 추적이 어려움
- 소프트웨어 안전 개발 프로세스 적용에 따른 일정 및 예산 부족
- 소프트웨어 안전 개발 프로세스를 이해하는 전문가 부족

비표준 분야에서는 다음과 같은 문제점이 도출되었다.

- 소프트웨어 안전 제품 개발 시 안전 계획 수립이 어려움
- 소프트웨어 안전 개발 시 요구사항 도출 및 요구사항 추적이 어려움
- 소프트웨어 안전 개발 프로세스의 각 단계별 기법과 산출물 관리 미흡

- 소프트웨어 위험원 관리 및 위험성 평가 미흡
- 소프트웨어 안전 개발 프로세스 적용을 위한 환경 부족
- 소프트웨어 안전 개발 프로세스 적용에 대한 경험 부족
- 소프트웨어 안전 개발 프로세스를 이해하는 전문가 부족
- 소프트웨어 안전 개발을 위한 소프트웨어 공학 기술에 대한 이해 부족

이러한 문제점 해결을 위해 표준 분야에서 다음과 같은 개선 방안을 도출하였다.

- 소프트웨어 안전 개발 프로세스에 대한 지속적인 교육
- 소프트웨어 안전 개발 프로세스를 위한 전문 인력 확보
- 소프트웨어 안전 개발 프로세스에 대한 가이드나 방법론 제공
- 소프트웨어 안전 개발 프로세스 적용하기 위한 제도 장치 마련

비표준 분야에서는 다음과 같은 개선 방안을 도출하였다.

- 소프트웨어 안전 개발 프로세스에 적용을 위한 인식 개선 및 교육 수행
- 소프트웨어 안전 개발 프로세스를 수행 할 수 있는 인력 확보
- 소프트웨어 안전 개발 프로세스에 대한 표준 개발 및 가이드 제공
- 소프트웨어 안전 개발 프로세스 적용하기 위한 법제도 장치 마련

5. 정책적 활용 내용

본 연구결과는 소프트웨어 안전 개발 프로세스 적용을 위한 국내외 표준 및 사례를 조사하고 안전 표준이 있는 분야와 없는 분야의 실무자들의 적용 현황 및 기대사항을 조사하여 소프트웨어 안전 프로세스 확산을 위한 방안 및 정책 마련에 기초 자료로 활용될 수 있다.

6. 기대 효과

본 연구는 소프트웨어 안전 개발 프로세스 적용 및 확산을 위한 정책 마련의 기초가 되며, 궁극적으로 국내 소프트웨어 안전 개발 역량을 강화하여 보다 안전한 소프트웨어를 만들어 관련 산업의 국제 경쟁력 확보에 기여할 것이다.

SUMMARY

1. Title:

Study on Activation of Development Process for Software Safety Assurance

2. Purpose and Necessity of the Research

As software dependency and complexity increase in each industry, the possibility of accidents caused by defects of software is increased. In case of accidents caused by software, social and economic damages can be enormous. Software safety is a key element for ensuring national safety. Has been highlighted. As software safety becomes more important, it is necessary to study how to implement it.

Software safety Advanced countries implement system safety standards together with software safety standards and ensure software safety. The main activity of the software safety standard is the safety process. The need for a policy to investigate and activate the activation of the software safety process has been raised.

The purpose of this study is to examine the application status of software development process and to find out how to apply and spread software safety development process according to product and company situation.

3. Composition and Range

In Chapter 1, we present the background, research purpose, content and method of conducting the research.

In Chapter 2, we describe the elements for software safety. We organize software development processes, software safety processes, and examples for safety processes.

In Chapter 3, we organize the survey paper for the actual condition of the software safety development process.

In Chapter 4, we summarize the findings of the survey and in - depth interviews and analyze the results of the survey on the application of the safety process.

In Chapter 5, we analyze the problems and suggest ways to improve software safety.

In Chapter 6, we describe the summary of the study, future research, and limitations of the study.

4. Main Contents and Results

In this study, we investigated the background knowledge of international standard software safety necessary to develop a proposal for applying the software safety development process. We have investigated the basic concepts of software safety and defined activities that are basically carried out to ensure safety. To ensure software safety, the first safety function needs to be implemented. Second, safety integrity according to the safety level must be ensured. Finally, software should be developed in accordance with the safety development process. The difference between the safety development process and the existing software development process is the addition of risk analysis and safety verification.

The survey was divided into non-standard and standard fields of automobile, aviation and railway. For the construction of the safety process application survey paper, the software safety process was defined in the standards of IEC 61508, ISO 26262, IEC 62279, and DO-178C. We discussed with the Software Safety Advisory Group and confirmed the facts and questions for applying the software safety development process.

The questionnaire consisted of the surveyed companies' status, common questionnaires on standard and non-standard fields, questionnaire on each field of software safety process, future improvement and support requirements. For each field survey, we set up common requirements analysis, architecture design, detailed design, implementation and testing. The common questions about whether or not to perform each step were constructed, and the contents of the safety development process with each industry characteristic were added as sub - questions. The standard field is DO-178C for aeronautical field, ISO 26262 for automobile field, and IEC 62279 for railway. In the case of non-standard fields, the study item was designed based on IEC 61508, the functional safety standard.

We conducted an in-depth surveys to find out what needs to be done to apply the software safety process to spread the software safety industry. A comprehensive analysis of the common questionnaire on software safety generalities showed that the railway and aviation sectors showed a high level of consciousness, preparation and

activities for safety. The non-standard areas were found to lack awareness, knowledge, and performance in terms of risk analysis and safety management.

Based on the analysis of the survey results, the following problems were found in standard fields such as automobile, aviation, and railway.

- Difficulty in applying techniques for software safety assurance
- Sufficient design is not provided to implement software safety in detail design process
- lack of detailed process information at each stage of software safety development process
- The output of the test phase is difficult to trace software safety requirements against the plan
- Schedule and budget shortage due to application of software safety development process
- Lack of experts to understand the software safety development process

In the non-standard field, the following problems were derived.

- Difficult to establish safety plan when developing software safety product
- Difficulty in deriving requirements and tracking requirements in software safety development
- Inadequate management of each step of the software safety development process
- Lack of management of each state of technique and outcome of software safety development process
- Lack of environment for applying software safety development process
- Lack of experts to understand the software safety development process
- Lack of understanding of basic software engineering skills for software safety development

In order to solve these problems, the following improvement measures have been derived in the standard field.

- Continuous training on the software safety development process

- Gaining experts for software safety development process
- Providing guidance and methodology for software safety development process
- Establishing a system for applying the software safety development process

In the non-standard field, the following improvement measures were derived.

- Improving awareness and training for application to the software safety development process
- providing that personnel are able to perform the software safety development process
- Developing standards and guide for software safety development process
- Providing legal system for applying software safety development process

5. Policy use

In this study, we investigate domestic and international standards and cases for application of software safety development process. We investigate the application status and expectation of practitioners in fields with and without safety standards. The results of this study can be used as basic data for policy proposal and policy making process.

6. Research Implication and Expected Effects

This study is the foundation of the policy for applying and spreading the software safety development process. Ultimately, it will contribute to obtaining the international competitiveness of related industries by strengthening domestic software safety development capability and creating safe software.

CONTENTS

제1장 서 론

제1절 연구 배경과 필요성

제4차 산업혁명은 소프트웨어 관점으로 볼 때 세 가지 특징이 있다. 첫 번째 특징은 타산업과 ICT(Information and Communication Technology)산업과 융합하여 편리하고 새로운 가치를 주는 서비스가 창조되었다는 점이다. 이 서비스는 CPS(사이버물리시스템, Cyber Physical System)[1]로 지원이 되며, CPS의 특징은 시스템 확장성(Scalability), 융복합성(Composability), 상호작용성(Interactivity), 고신뢰성(Dependability), 실시간성(Timing), 상호운용성(Interoperability), 지능성(Intelligence)로 복잡하고 지능적이며 안전이 중요한 시스템이다.[2] 두 번째 특징은 그 서비스 창출을 위해 소프트웨어가 하드웨어를 제어하는 기능이 증가되었다는 것이다. 다양한 스마트센서를 통하여 물리 세계의 정보를 수집하고 분석한 결과를 작동장치(Actuator)를 통하여 물리 세계의 대상 시스템에 즉시 적용가능하다. 가장 대표적이고 활발하게 연구되고 있는 시스템이 무인자동차로 레이더, 라이더(LIDAR, Light detection and ranging), 카메라 등으로 주위의 환경과 물체를 인식하고, 목적지까지 경로를 설정하여 자율적으로 주행한다. 마지막으로 자동차, 항공, 철도 등을 포함한 기존 안전 산업에 소프트웨어가 큰 역할을 함으로서 소프트웨어안전[3]이 산업의 안전에 중요한 역할을 하게 되었다는 점이다.

세 가지 특징을 반영한 사례로 도시철도 열차제어시스템은 수동운전에서 자동운전으로 소프트웨어가 열차를 제어하는 방향으로 진행되고 있으며, 국내에서도 열차제어 소프트웨어를 개발하고 있다. 이러한 소프트웨어를 장착한 시스템이 상용화되기 위해서 가장 중요한 요소 중의 하나가 소프트웨어의 안전 확보이다.

---

1) 컴퓨팅 환경에서 물리적 환경 정보(데이터) 처리 결과로 물리 세계, 시스템 또는 프로세스를 제어하는 고신뢰 시스템(Dependable Systems)으로, 예로는 항공, 자동차를 포함하여, 스마트 제조, 스마트 헬스, 스마티 시티 등에 적용할 수 있다.

2) 진희승(2017), CPS의 기능, 비기능적 SW요구사항 분석, SPRi

3) 소프트웨어로 인한 사람의 생명이나 신체에 대한 위험의 발생을 방지하거나 이에 대한 충분한 대비가 되어 있는 상태, 소프트웨어산업진흥법 재・개정안, 2017.12

[그림 1-1] 도시철도 열차제어시스템의 발전

출처 : 황종규(2017) , 철도분야에서의 SW안전, SW안전 국제컨퍼런스

KRTCS (Korean Radio-based Train Control System)

그 동안 크게 부각되지 않던 소프트웨어안전이 중요해짐에 따라 그를 구현할 수 있는 방법에 대한 연구가 필요하게 되었다. 소프트웨어안전 선진국은 시스템 안전 표준과 함께 소프트웨어안전 표준을 정하고, 소프트웨어안전을 지키기 위한 활동을 하고 있다. 소프트웨어안전 선진국은 소프트웨어 안전 표준을 무역 장벽으로 이용하며, 안전 제품의 수출 등을 위해서는 안전 표준을 준수가 필요하다.

2015년과 2016년에 소프트웨어정책연구소에서 수행한 『소프트웨어안전 산업 동향 조사』에서는 국내의 경우 소프트웨어안전 프로세스 활동 중에서도 가장 중요한 위험도 분석이나 안전 메카니즘 분석, 설계보다는 테스트위주의 활동을 위주로 활동한다고 조사되었다. 2018년의 『소프트웨어안전 산업 동향 조사』에는 위험도 분석이나 안전 메카니즘 분석이 중요한 활동으로 나와 소프트웨어 안전에 대한 개념이 정착되었으나, 여전히 안전 프로세스 적용에는 인력, 비용 등 어려움이 있는 것으로 조사되었다. 또한 개발 프로세스 적용 및 문서화에 대한 문화가 정착되어 있지 않아 IEC 61508, ISO 26262, DO-178C 등 프로세스 기반의 안전 표준 적용에 어려움을 토로하고 있다.

이에 소프트웨어안전 확보를 위한 소프트웨어 개발 프로세스 적용 현황을 면밀히 확인하고, 제품 및 기업 현황에 맞는 소프트웨어 안전 개발 프로세스의 적용과 확산 방안 연구가 필요하다.

제2절 연구 목적

소프트웨어안전 표준은 IEC 61508[4]이라는 기능안전 표준을 기반으로 자동차, 항공, 철도 등의 표준으로 분화되었다. 하드웨어의 안전은 안전 무결성 수준(Safety Integrity Level)을 결정하고 그에 맞는 실제 고장률을 측정하여 그 달성도를 확인한다. 소프트웨어안전의 경우는 실재 고장률 측정이 불가능하기 때문에 소프트웨어안전을 위한 개발 프로세스를 정하고 이에 따라 개발한 경우 소프트웨어안전이 구현되는 것을 전제로 하고 있다. 그러나 안전 프로세스 적용 시 프로세스의 복잡성과 프로세스에 의한 안전 확보에 대한 불확실성이라는 문제점이 있었다.

소프트웨어안전 선진국에서는 안전에 대한 새로운 표준 검토 필요성이 나오고 있다. 미국 국회에 제출된 자료는 새로운 자동차 표준에 대한 검토사항을 포함하고 있다. 안전이 중요해지는 중의 하나로 소프트웨어의 복잡성의 증대로 테스트가 불가능하고, 빠른 기술의 변화를 들고 있다. 복잡한 시스템에서 안전 확보를 위해 모든 조건을 검사하려면 시간과 비용을 낭비가 심해, 새로운 안전 프로세스 표준이 필요하다고 주장한다.[5]

이에 본 연구의 목적은 국내 소프트웨어 안전 개발 프로세스 적용 현황을 조사하고 확산 방안을 마련하는데 있다. 정책 당국에는 소프트웨어 안전 확보 정책 방향 제시를 위한 근거 자료를 제공한다. 항공, 철도, 자동차, 원자력 등 안전 중요 시스템은 물론이고 소프트웨어가 제어하는 시스템 구현 시 소프트웨어 안전 관련 기업에 안전 소프트웨어 개발 프로세스 적용의 중요성에 대한 인식을 제고하고, 프로세스 적용에 필요한 자료를 제공한다.

---

4) Functional Safety of Electrical/Electronic/Programmable Electronic Safety-related Systems

5) NHTSA(National Highway Traffic Safety Administration, 2015), Report to Congress: “Electronic Systems Performance in Passenger Motor Vehicles”, 2015.12. , pp44

제3절 연구 내용

본 연구의 내용은 다음과 같다.

소프트웨어 안전 프로세스 적용에 대한 소프트웨어 안전 선진국의 사례 조사 및 소프트웨어 안전 프로세스 조사를 위한 항목, 수준, 대상 선정 등 사전 작업을 수행한다. 안전 프로세스 조사를 위한 항목은 가능한 조사 대상자들이 이해하기 용이하게 조사지를 구성한다. 특히 비표준 분야의 조사지는 IEC 61508을 기준으로 작성하되, 표준을 잘 모르는 조사 대상자들도 이해 가능한 용어로 변경하여 조사지를 구성한다.

소프트웨어 안전 표준이 있는 경우, 프로세스 적용 수준, 프로세스 미적용 시 이유에 대해 집중적으로 인터뷰하고 적용을 위해 지원해야 하는 사항을 도출한다.

또한 소프트웨어 안전 표준이 없는 분야는 조사 대상자의 소프트웨어 안전 개념을 확인하고 그 사업군에서 안전을 위해 일반적으로 수행하는 활동을 조사하여 소프트웨어 안전 확보 방법을 도출한다.

소프트웨어 안전 선진국의 사례 조사 및 안전 관련 기업의 심층 인터뷰를 통해 도출된 결과를 바탕으로 안전 프로세스 확보방안을 도출한다.

제4절 연구 방법

본 연구는 체계적인 소프트웨어 안전 개발 프로세스 적용 실태에 대한 최초의 연구로 국내외의 안전 표준, 관련 사례 및 자료 조사가 선행되었다. 문헌 연구와 사례 조사를 통하여 국내외 소프트웨어 안전 개발 프로세스 미적용 시 사고 사례와 적용 우수 사례도 조사하여 정리하였다.

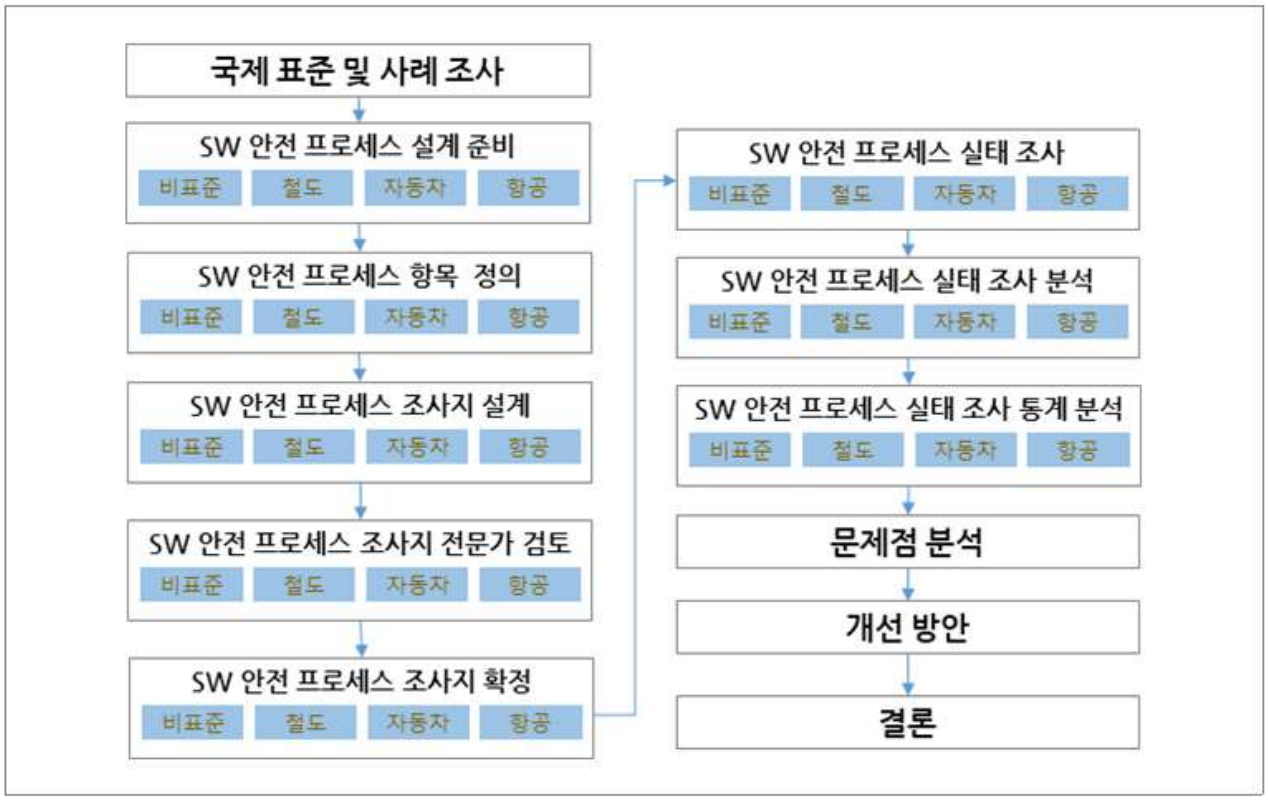
소프트웨어 안전 개발 프로세스는 소프트웨어 품질 프로세스가 기반이 되어 발전된 것이다. 선행 연구 및 각종 자료를 바탕으로 한 문헌 연구를 통해 소프트웨어 안전뿐만 아니라, 소프트웨어 품질에 대한 연구도 검토하여 소프트웨어 안전 개발 프로세스 적용에 필요한 조사 항목을 도출하였다. 본 연구에서는 가

능한 품질 프로세스는 제외하고 소프트웨어 안전 개발 프로세스 위주로 조사하였다. 그러나 안전 프로세스에 반듯이 포함되어야 하는 품질 프로세스는 포함하였다. 국제 소프트웨어 안전 표준 전문가와 소프트웨어 안전 관련 전문 위원들의 자문을 통하여 소프트웨어 안전 개발 프로세스 적용 실태조사를 위한 조사 항목을 검증하고 정제하였다.

국제 안전 표준이 있는 산업 분야와 비표준 산업 분야의 소프트웨어 안전 개발을 담당하는 현장 실무자에 대한 조사를 수행하였다. 자동차, 항공, 철도 등의 안전 표준이 있는 분야와 조선, 제조 등 안전표준이 없는 분야 기업이 조사 대상이다.

자료 조사 및 전문가 검증으로 확정된 조사지 조사와 심층인터뷰를 통해 소프트웨어 안전 개발 프로세스 적용에 관한 산업 분야별 실태 조사를 하고 그 결과를 분석하였다. 분석된 결과를 바탕으로 소프트웨어 안전 개발 프로세스 확산을 위한 방안을 도출하였다.

[그림 1-2] 연구 방법 도식화



제2장 소프트웨어 안전 프로세스

제1절 소프트웨어 안전 프로세스 개요

1. 안전에 대한 이해

안전성에 대한 정의는 국제 안전규격을 위한 가이드인 ISO/IEC GUIDE51[6]에 기술되어 있다. ISO/IEC GUIDE51에서 안전성은 '수용할 수 없는 위험성이 없는 것(freedom from unacceptable risk)' 이라고 표현되어 있다. 직역하면, '수용할 수 없는 위험으로부터의 해방' 이라고 정의하고 있다. 안전성이란 사람 또는 재산에 대한 재해의 위험성이 허용 가능한 수준으로 억제되어 있는 상태라 할 수 있다.

Storey(1996년)는 안전에 대한 개념을 시스템 측면에서 1차적 안전(Primary safety), 기능 안전성(Functional safety), 간접 안전성(Indirect safety)으로 정의했다. 1차적 안전성은 하드웨어의 화재, 감전 등에 의한 직접적 사고로부터의 안전성이며, 기능 안전성은 시스템의 위험 분석 결과에 따라서 안전 메카니즘 구현을 통해 위험이 제거되는 시스템의 안전성이며, 간접 안전성은 잘못된 정보 제공으로 일어날 수 있는 위험원에 대한 안전성을 말한다.[7]

안전성(Safety)은 확률적인 개념으로 절대적 안전이란 존재하지 않고 상대적인 안전 정도로 판단한다. 안전성 평가는 확률 또는 정도의 개념으로 위험성을 정량화하여 위험성을 제거 또는 감소하기 위한 기능이나 대책을 마련하는 위험성 평가 과정을 포함하며 여기서 나온 기능 및 대책이 안전 기능 및 안전 대책이 된다.

시스템이나 장비에 의해 좌우되는 전체적인 안전성의 일부분으로 안전 기능을 수행하는 시스템이나 장치는 조건 입력에 대해 사전에 정의한 대로 의도된 반응이 올바르게 수행되어야 한다.

---

6) ISO/IEC Guide 51 (2014), "Safety aspects - Guidelines for their inclusion in standards"

7) Smith, David J. and Simpson, Kenneth G. L. (2004), "Functional Safety(A Straightforward Guide to Applying IEC 61508 and Related Standards)", Hutterwirth-Heinemann

2. 소프트웨어 안전에 대한 이해

1) 안전 기능

안전한 소프트웨어를 개발한다는 의미는 소프트웨어가 가지고 있는 여러 가지 속성 중에서 특히 안전 기능(Safety Function)을 올바르게 추출하였는지를 확인할 수 있어야 하고, 최종 개발 결과물에서 최초 의도한 요구사항대로 안전 기능이 정상적으로 구현되고 작동하는가를 확인하는 것이다.

안전 기능은 시스템이나 장비에 의해 좌우되는 전체적인 안전성의 일부분으로 안전 기능을 수행하는 시스템이나 장치는 조건 입력에 대해 사전에 정의한 대로 의도된 반응이 올바르게 수행되어야 한다. 안전한 시스템을 개발하는 단계는 제품의 기능 안전성을 확보하기 위해 안전 기능을 도출하는 단계와 이렇게 도출된 기능을 구현하는 과정에서 원하는 수준의 안전성을 확보할 수 있는 방안을 마련하는 단계, 이렇게 크게 두 가지 부분으로 나누어서 살펴볼 수 있다. 시스템 단계에서 도출된 안전 기능은 소프트웨어 단계로 상속되며, 소프트웨어 단계에서 구현된 안전기능은 시스템 단계에서 통합적으로 테스트되어야 한다.

2) 안전 무결성

시스템을 설계하는 초기 단계에서는 여러 가지 큰 위험성(risk)이 존재하고 안전상 불안한 상황이므로 각각의 위험성에 대하여 설계, 제작 및 운영 단계에서 각종 안전 기능 또는 안전 대책을 마련하여 위험성의 크기를 줄여야 한다. 시스템을 사용하는 누가 생각하더라도 이 정도 크기의 위험성만 존재한다면 문제가 되지 않는 상태인 수용 가능한 위험만을 남겨두고 나머지 위험성은 제거 또는 감소 시켜야 한다.

[그림 2-1] 위험성과 안전성 관계

출처 : 정보통신산업진흥원(2017), 『공통 분야 소프트웨어신뢰·안전성 확보를 위한 가이드』

위험성이 매우 적거나 적게 되었기 때문에 문제가 되지 않은 위험성 영역이 바로 진정으로 안전한 상태(Safety State)라고 말할 수 있다. 안전 무결성(Safety Integrity)은 주어진 모든 조건하에 있는 안전관련 시스템이 주어진 시간 내에 요구되는 안전기능을 만족스럽게 수행할 수 있는 확률이라 말할 수 있다.

안전기능 요구사항은 위험원 분석을 통해 도출되고, 안전 무결성 요구사항은 위험성 평가를 통해 도출되고 결과에 의해서 안전 무결성 수준(Safety Integrity Level)이 결정된다. 안전 무결성 수준이 높을수록 해당 장비 또는 시스템의 고장 발생 가능성은 낮아진다. 안전 무결성 수준이 널리 사용되는 이유는 적용 대상이 명확하게 정의되지 않은 경우에 시스템의 안전성을 평가하여 시스템의 안전성을 상호 비교할 수 있는 방법이기 때문이다.

3) 안전한 소프트웨어 개발의 핵심 요소

소프트웨어 개발 프로세스에 대한 접근은 안전이 중요시됨에 따라 소프트웨어 중심의 프로세스 관점에서 시스템과 소프트웨어를 동시에 바라보는 개념으로 변화하고 있다. 이와 같은 변화는 소프트웨어를 전체 시스템의 일부로 인식하여 전체 시스템 레벨에서 위험성 분석을 선행하고, 시스템 위험분석 결과로 도출된 시스템 안전요구사항을 바탕으로 소프트웨어 안전요구사항을 도출해야 한다는 것을 의미한다. 소프트웨어 안전성을 보증한다는 의미는 소프트웨어가 가지고 있는 여러 가지 속성 중에서 특히 안전기능(Safety Function)이 올바르게 선정되었는지를 확인할 수 있어야 하고, 최종 개발 결과물이 수행하는 안전기능이 정상적으로 작동하는가를 확인하는 것이다.

안전한 소프트웨어 개발이라는 목표를 달성하기 위해서는 다음의 세 가지가 가장 핵심적인 요소라 할 수 있다[8].

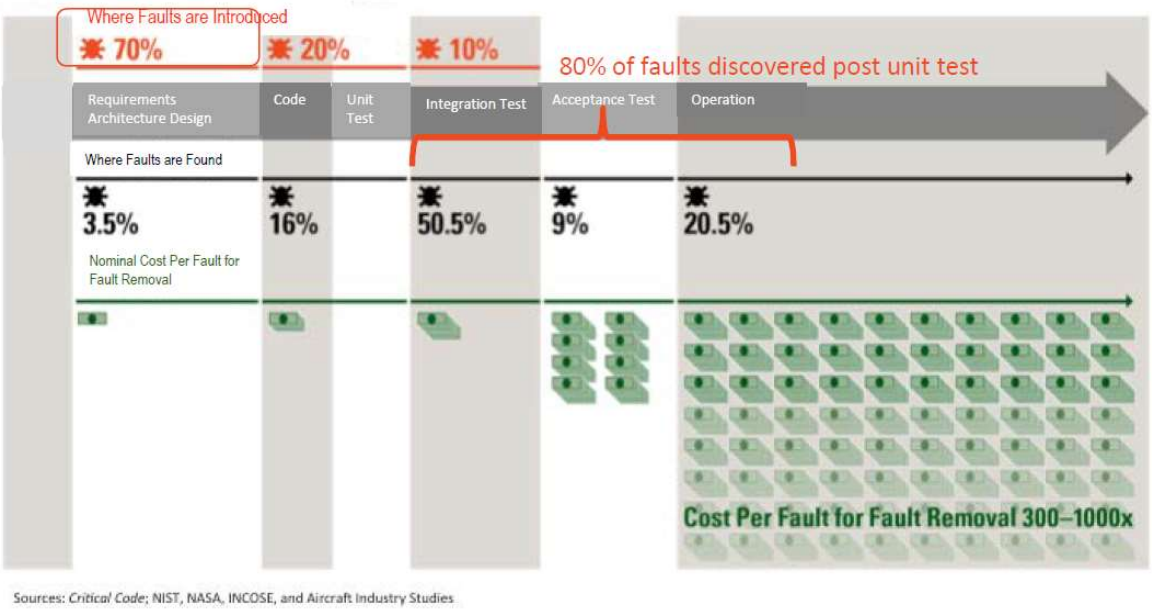
첫째, 소프트웨어 개발 안전 수명주기에 대한 이해이다. 소프트웨어 개발 안전 수명주기는 개발 조직 내의 모든 개발자가 숙지하고 각 단계 별 결과물들을 사전에 계획된 바에 따라 생성해 나가야 하는 것인데 수명주기는 경우에 따라 국제표준에서 제시하는 가이드를 일부 수정하여 개발 프로젝트에 최적화시킬 수 있다.

이때 안전 무결성 수준을 높이고 비용을 줄이기 위해서는 개발 앞단인 위험 분석 단계에 집중해야 한다. 일반적인 소프트웨어 개발 단계와 동일하게 안전 소프트웨어 개발 시에도

8) 정보통신산업진흥원(2017), 『공통 분야 소프트웨어신뢰・안전성 확보를 위한 가이드』

이 법칙이 적용된다.

[그림 2-2] 안전 중요 소프트웨어 개발 단계 및 비용



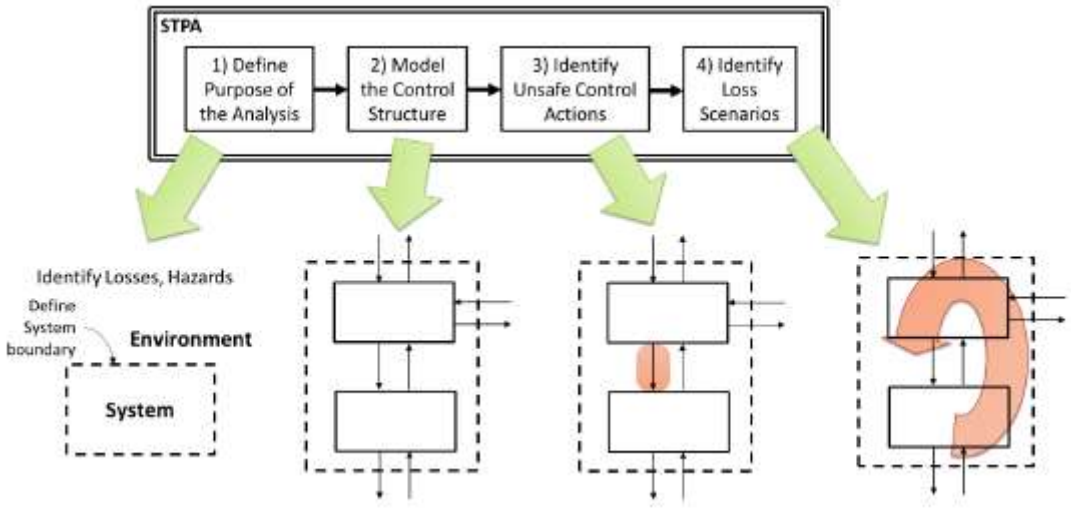
출처 : Peter Feiler, Challenges in Existing System Safety Practices, CMU, Software Solutions Symposium 2017

둘째, 시스템 및 소프트웨어 위험분석 기법에 대한 이해가 필요하다. 안전한 소프트웨어 개발은 시스템 위험분석을 통하여 도출된 시스템 안전기능 요구사항을 근간으로 개발이 이루어지기 때문이다. 시스템 레벨에서 도출된 안전기능 요구사항 중에서 소프트웨어가 수행하여야 하는 요구사항들이 소프트웨어 관점에서 명세화되고 이를 바탕으로 소프트웨어가 구현되어야 한다.

소프트웨어 위험분석 기법으로는 SFMEA(Software Failure Modes and Effects Analysis), Software Fault Tree Analysis (SFTA) 등이 있다. 이러한 분석 기법은 선형적인 분석기법으로 현재 복잡한 시스템의 위험을 분석하기는 적합지 않아, 시스템 요소들간의 관계를 분석하여 위험요소를 찾아내는 STPA(System Theoretic Process Analysis)가 개발되고 보급되고 있다.[9] 기본적인 방법은 분석의 목적 정의, 분석 대상인 시스템 모델링, 위험한 조절 행동 추출, 손실 추출 및 방지라는 4단계가 적용되며, 간단하지만 시스템에 적용하여 분석 시 유용하고 효과적인 방법이다.

9) Nancy G. Leveson et al.(2018) , STPA Handbook

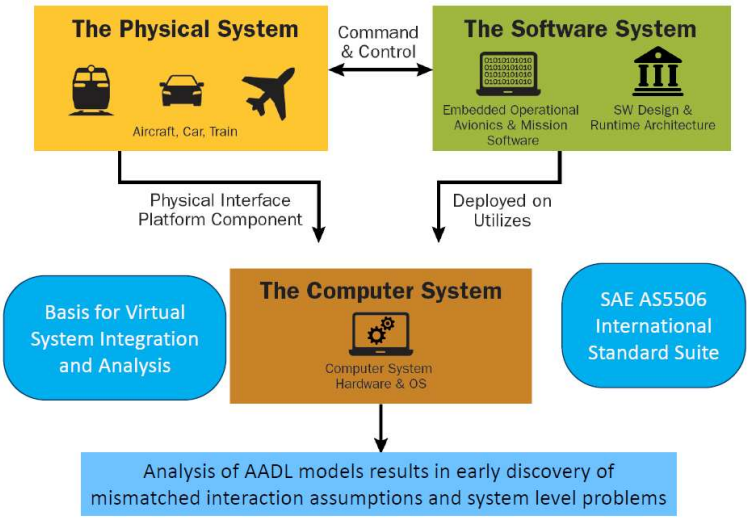
[그림 2-3] 기본 STPA 적용 방법 개요



출처 : Lancy G. Leveson et al.(2018) , STPA Handbook

위험분석은 노력과 비용이 많이 들어가는 작업이기 때문에 자동화에 대한 연구가 일찍부터 시작되었다. 위험 분석을 기반으로 하여 안전시스템 설계와 구현이 이루어지기 때문에 위험분석이 자동화되면 설계와 구현에 제대로 구현되었는지 검증절차도 용이하게 진행할 수 있다. AADL(Architecture Analysis & Design Language)을 이용하여 전체시스템, 소프트웨어, 하드웨어간의 상호 작용을 분석하여 시스템을 디자인하고 검증하는 연구가 있다.[10)]

[그림 2-4] SAE AADL 표준



출처 : Peter Feiler, Virtual System Integration and Verification, CMU, Software Solutions Symposium 2017

10) Myron Hecht, Aerospace Corp., member of AADL & DO-178C committee, Automated safety analysis of several satellite systems for JPL (2009-)

셋째, 소프트웨어 안전기능이 의도한 바와 같이 구현이 되었는지를 확인하는 올바른 검증 시험 방법을 적용하는 것이 중요하다. 소프트웨어의 안전성을 검증하는 시험은 디바이스 레벨에서 이루어지며, 검증의 목표는 시스템 안전 요구사항 및 소프트웨어 안전 요구사항이 원래 의도와 같이 구현되었는지를 검증하는 것이기 때문이다. 안전 표준에서는 각 수명주기별 수행해야 할 안전 무결성 수준별 기술/평가, 방법, 목표 등을 제시하고 있다.

안전 선진국에서는 안전 프로세스의 적용에 추가하여, 시스템의 안전 확보 검증을 위해 안전진술(Safety Case)을 활용하고 있다. 안전진술은 특정 운용 환경에서 안전 검증을 위해 위해도 분석 결과와 검증 결과를 증거(Evidence)로 하여 논증(Argument)을 작성한다. 안전진술은 시스템 개발 시 안전을 고려하면서 개발하는 데도 사용되며, 안전 검증자가 안정성 평가 시에도 활용이 가능하다. ISO 26262에서도 안전진술을 요구하고 있다. 안전진술 작성은 Goal Structuring Notation (GSN)을 사용하여 논증(Argument)을 전개할 수 있다.[11)]

11) 권기춘 외(2017), 원전 안전 소프트웨어 위해도 및 안전보증 평가기술 개발, 한국원자력연구원

3. 소프트웨어 품질 프로세스와 비교

1) 소프트웨어 품질 프로세스의 개요

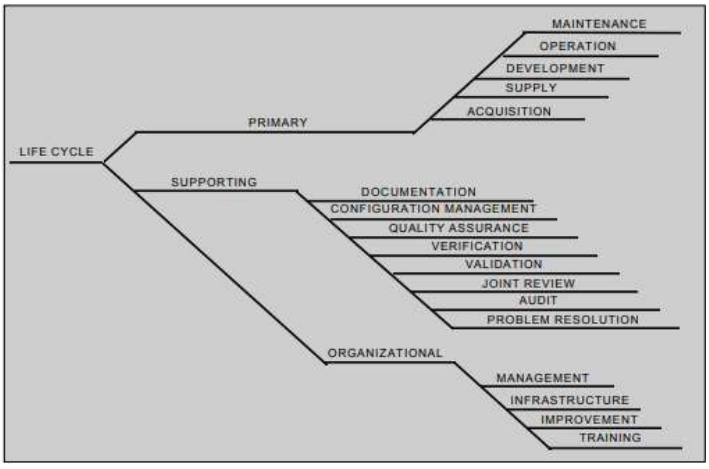
산업 전반에 걸쳐 소프트웨어의 사용이 증가함에 따라 소프트웨어 개발 프로세스 표준도 이에 발맞추어 지속적으로 발전되어 왔다. 소프트웨어 개발 프로세스 측면에서는 ISO/IEC 12207을 다양한 조직 및 분야에서 소프트웨어 개발 방법론의 기본적인 참조 모델로 적용하고 활용하고 있다. 국내에서도 ISO/IEC 12207을 기반으로 국가 소프트웨어 발주관리 프로세스 표준으로 정의하여 사용되고 있다[12].

2) ISO/IEC 12207 프로세스 구성

ISO/IEC 12207은 소프트웨어 도입을 위한 전체 수명주기 과정에 있어서의 일련의 기본적인 활동과 이를 수행하기 위한 프로세스상의 절차 및 작업내용 등을 관리적, 공학적 측면에서 규정한 상위 수준의 틀이다.

ISO/IEC12207의 프레임워크는 핵심(Primary), 지원(Supporting), 조직(Organization) 수명주기 프로세스 등 3개의 수명주기 프로세스 그룹과 세부프로세스로 구성되어 있으며 각 프로세스는 다수의 활동으로 구성된다. 수명주기 프로세스는 각각의 하위 프로세스의 업무가 기능적으로 응집성이 높고 타 프로세스와의 인터페이스가 최소화된 모듈화(Modularity) 구조이다.

[그림 2-5] 수명주기 프로세스 및 하부 프로세스 (17개)



출처 : Raghu Singh, INTERNATIONAL STANDARD ISO/IEC 12207 SOFTWARE LIFE CYCLE PROCESSES, Federal Aviation Administration

12) 한국소프트웨어진흥원 (2005), 『시스템 수명주기 프로세스 프레임워크 보고서』

ISO/IEC 12207은 소프트웨어 개발 및 관리에 적용될 수 있는 공정(17개 Process), 활동(74개 Activity) 및 세부 활동(224개 Task)을 정의하고 있다. 향 루 안전 프로세스와 비교하기 위해 핵심 프로세스 중 소프트웨어 개발, 소프트웨어 지원 영역에 대해 확인한다.

(1) 소프트웨어 개발

소프트웨어 개발 영역은 소프트웨어 요구분석, 소프트웨어 구조 설계, 소프트웨어 상세 설계, 소프트웨어 구현 및 단위시험, 소프트웨어 통합 시험, 소프트웨어 자격시험 등의 프로세스로 구분되며, 이 프로세스들은 반복되거나 중복될 수 있다.

소프트웨어 요구분석은 개발하고자 하는 소프트웨어의 요구사항을 분석하고, 이에 대한 검토 및 평가를 수행하고, 정의된 소프트웨어 요구사항이 사용자 요구사항에 부합되는지 검토하고 확정하는 것이다.

소프트웨어 구조 설계는 시스템 구성항목에서 소프트웨어의 구성 요소를 식별하고, 이들 간의 관계를 정의하는 활동이다. 소프트웨어 구조는 전체 시스템에 대한 기능 및 성능에 영향을 줄 수 있어 시스템의 특성에 따라 정의해야 한다. 개발자는 소프트웨어 통합시험 계획을 작성한다.

소프트웨어 상세 설계는 구조 설계에서 식별된 소프트웨어 구성요소로부터 단위 소프트웨어를 정의하고 단위 소프트웨어 알고리즘, 인터페이스를 설계하는 활동이다. 개발자는 소프트웨어의 단위시험 계획을 작성하고, 소프트웨어 통합 시험 계획을 보완한다.

소프트웨어 구현 및 단위시험 단계에서는 소프트웨어 상세설계 내역을 이용하여 단위 소프트웨어 및 데이터베이스에 대한 코딩을 수행하고, 소스코드를 생성한다.

소프트웨어 통합 및 시험 단계에서 개발자는 소프트웨어 통합계획을 작성하고, 계획에 따라 소프트웨어를 통합하고 결과를 시험한다. 개발자는 소프트웨어 자격시험을 위한 요구사항 및 절차를 정의한다.

소프트웨어 자격시험 시 개발자는 소프트웨어 항목에 대한 자격 요구사항에 따라 소프트웨어 자격시험을 준비하고 계획에 따라 자격시험을 수행하고 결과를 검토한다.

각각의 소프트웨어 개발 단계에서 개발자는 기술 및 관리검토를 위해 발주자와 합동 검토 활동을 수행한다.

⑵ 소프트웨어 지원

소프트웨어 지원 프로세스는 문서화, 형상관리, 품질보증, 검증, 확인, 검토, 감리, 문제해결 등이 활동으로 구성된다.

소프트웨어 개발 프로세스에서 문서화 활동은 다른 수명주기 프로세스나 활동에 의하여 생산되는 정보를 기록하는 것으로, 이 단계에서 생성된 문서는 개발 코드와 함께 중요한 개발 산출물의 하나로 유지보수 및 업그레이드 등에 사용된다. 형상관리 프로세스는 다른 수명주기 프로세스의 필수 구성 부분이며, 개발 시스템 내의 소프트웨어 항목 식별과 정의, 수정 과 배포 통제, 수정 요구의 보고 및 기록 등이 포함되어야 한다. 품질보증은 사업기간 동안의 소프트웨어 산출물과 사업에 적용되는 수명주기 프로세스가 계약에 명시된 요구사항을 따르고 있고, 정의된 계획을 준수하는지를 보증하기 위한 프로세스이다.

검증 프로세스는 해당 프로세스 활동 결과인 소프트웨어 산출물이 이전 활동에서 명시한 요구사항과 상태를 충족하고 있는지를 결정하는 활동이다. 비용과 성능의 효과성을 제고하기 위하여, 검증 프로세스는 가능한 한 초기에 실행해야 한다. 확인 프로세스는 요구사항과 최종적으로 개발된 시스템 또는 소프트웨어가 명시된 요구사항과 상태를 만족시키는지를 결정하는 활동이다. 검토 프로세스는 사업 수행상태와 산출물을 평가하는 활동으로 사업관리 및 기술수준에서 이루어지며, 계약기간 동안 적용된다. 검토 프로세스는 발주자와 공급자 조직 모두가 참여하여 수행한다. 감리 프로세스는 사용자 요구사항, 사업계획, 계약에 대한 준수 여부를 결정하는 활동이다.

문제해결 프로세스는 소프트웨어 수명주기 동안에 발생하는 문제를 분석하고 해결하는 활동으로 필요에 따라 사업수행 책임을 갖는 상위 관리자가 직접 수행하기도 한다.

3) 품질 프로세스와 안전 프로세스의 비교

앞에서 설명한 소프트웨어 품질 프로세스의 표준이라 할 수 있는 ISO/IEC 12207과, 대표적인 소프트웨어 안전 표준과의 비교를 통해 소프트웨어 개발 프로세스의 공통점과 차이점을 알아보고자 한다.

비교 대상이 되는 안전 표준은 국제 기능 안전 모표준이라 할 수 있는 IEC 61508 표준에서 소프트웨어 개발 프로세스를 다루는 Part3, 항공분야 소프트웨어 안전 개발 표준인 DO-178C, 자동차 분야의 안전 표준인 ISO 26262에서 소프트웨어 안전 개발 프로세스가 제

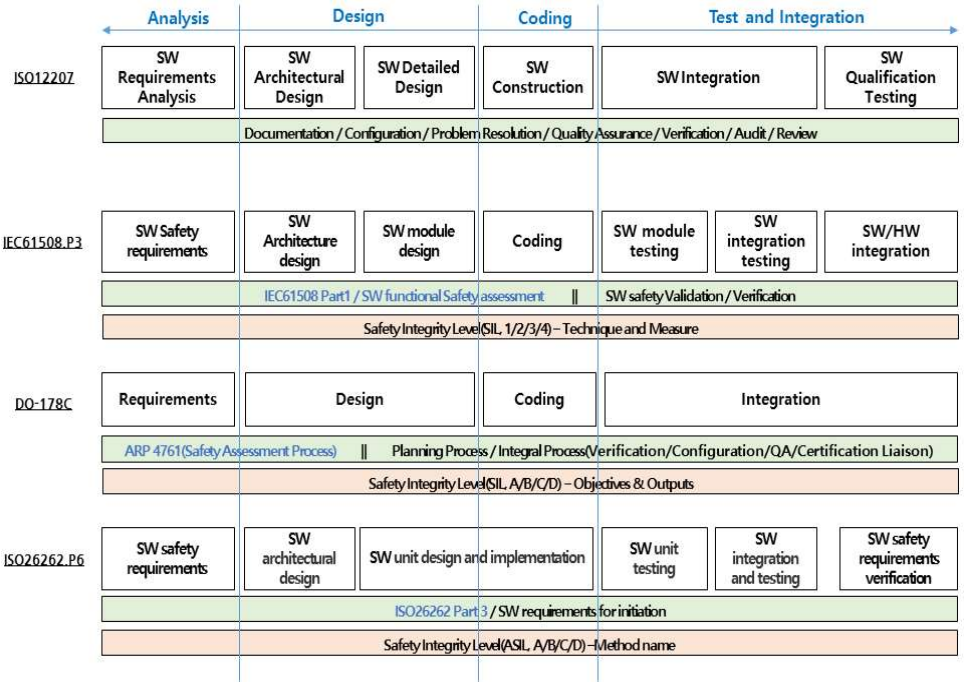
시되어 있는 Part6을 중심으로 비교하고자 한다.

소프트웨어 개발 절차는 오랜 기간 동안 발전해 오면서 상호 참조를 통해 향상되어 소프트웨어 개발 품질 프로세스와 안전 프로세스간의 기본적인 구조는 유사하다. IEC 61508 표준에서도 소프트웨어 개발 프로세스는 ISO/IEC 12207을 참조하게 되어 있다.

가장 큰 차이는 안전한 시스템 개발을 위해 소프트웨어 개발에 들어가기 전에 시스템 관점에서 전체의 위험을 분석하고 평가하는 단계를 거치며 이를 통하여 안전 기능과 안전 무결성 수준이 도출되고 여기서 도출된 안전 기능이 개발 전 단계에서 안전 무결성 등급에 맞게 각 수명주기 단계별로 분석되고 설계되고 구현되고 통합되고 있는지에 대한 보증활동이 추가된다. 인증을 받고자하는 분야의 안전 표준에서 제시하는 안전 무결성 수준별 기술과 평가, 방법, 목표에 대한 적용 또한 각 수명주기 단계별로 수행해야 할 내용이 된다.

아래 내용은 품질 프로세스 표준과 3개의 분야별 안전 표준간의 소프트웨어 개발 프로세스를 비교하여 제시한 것이다. 요구 분석 단계는 모두 동일하게 수행하지만 안전 표준에서는 앞 단계에서 위험 분석 및 평가 단계가 더 들어가 있다. 물론 설계, 개발, 테스트 과정에서도 위험 분석과정에서 나온 안전 관련 요구사항들을 설계하고 구현하여 안전을 확보하는 것이 필요하다.

[그림 2-6] 품질 프로세스와 안전개발 프로세스 비교 1

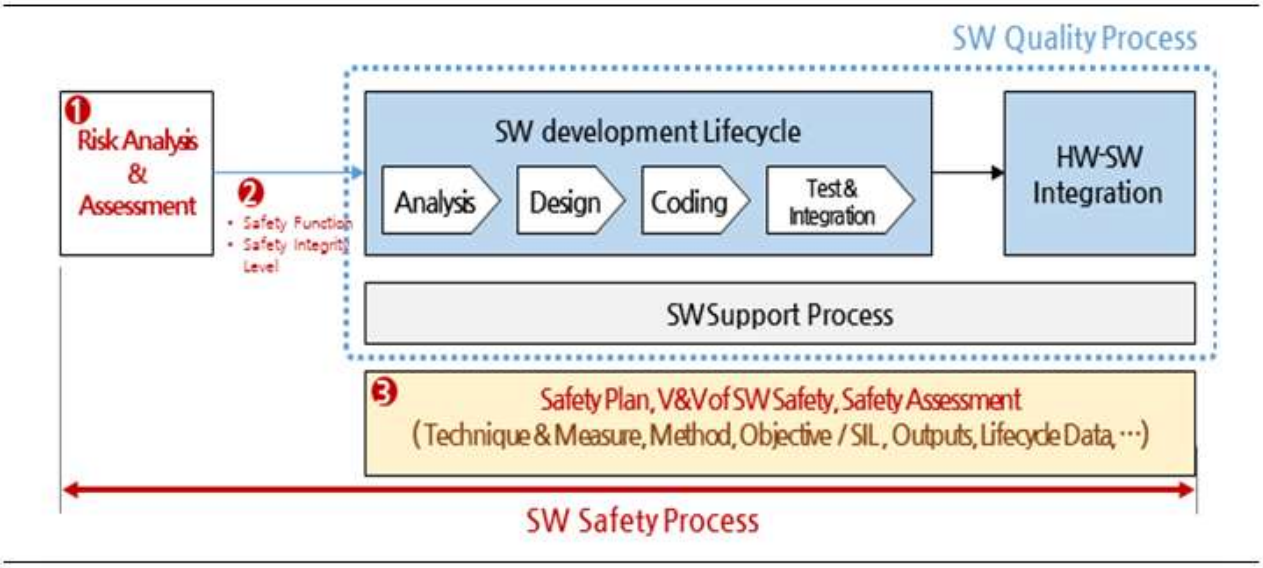


IEC 61508 에서는 Part1에서 소프트웨어 기능 안전 평가(SW functional safety Assessment)를 수행하고, DO-178C에서는 ARP 4761[13]에서 선행적으로 위험 평가를 하도록 규정되어 있고, ISO/IEC 26262 에서는 Part3에서 소프트웨어 위험 분석 및 평가를 수행하도록 되어 있다. 위험 분석 및 평가에 따라 도출된 소프트웨어의 안전 기능 및 안전 무결성 수준이 소프트웨어 개발 안전 요구사항으로 반영되어야 한다.

이후 소프트웨어 개발 수명주기 동안 수행하는 기본적인 활동은 품질 보증활동과 유사하다. 하지만 안전 표준에서는 안전 요구사항을 만족하기 위한 단계별 수행해야 할 활동, 기법, 산출물 등을 안전 무결성 수준에 맞게 제시되어 있다. 이와 같은 단계별 수행할 내용은 프로젝트 수행 초기에 결정된 안전관리 계획에 따라 실행되며 단계별 또는 프로세스 종료 시 안전성 보증에 대한 검토 및 확인을 수행해야 한다.

아래 그림은 앞서 비교한 그림을 소프트웨어 품질 프로세스와 안전 표준에 맞는 개발 프로세스로 구분하여 간략화한 것이다. 품질 프로세스보다 더 추가적으로 수행해야 하는 내용은 먼저 위험 분석 및 평가, 이를 통해 안전 기능과 안전 무결성 수준이 결정되고, 이에 맞도록 각 개발 수명주기에서 안전 보증 및 확인 활동을 수행한다.

[그림 2-7] 품질 프로세스와 안전개발 프로세스 비교 2

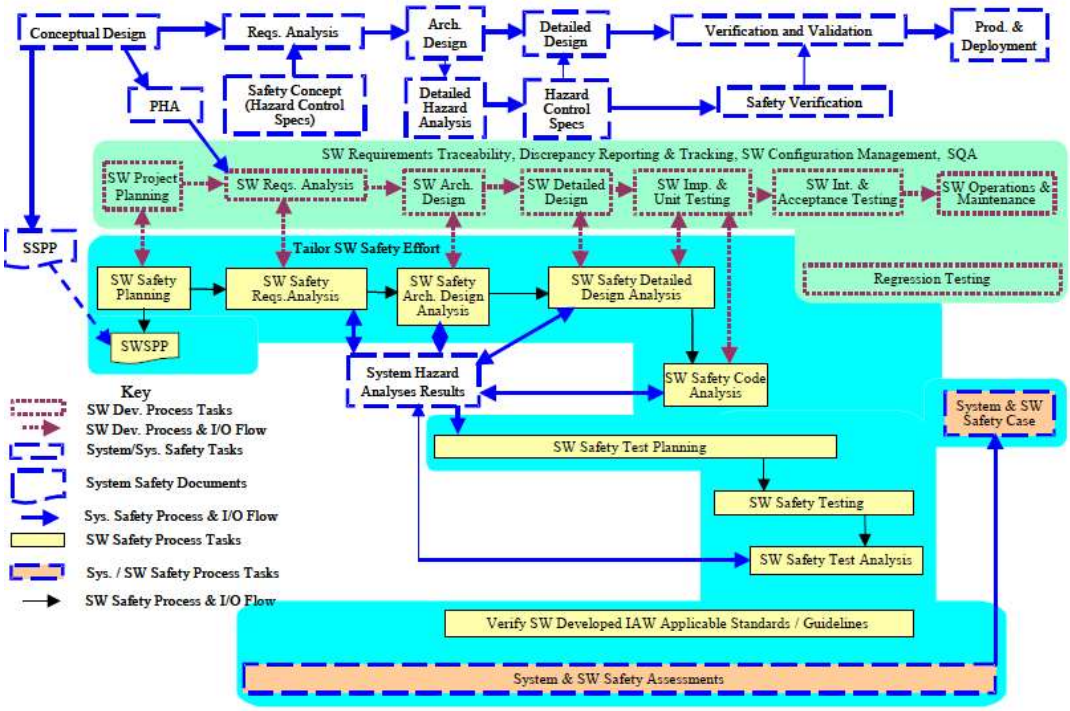


13) SAE(Society of Automotive Engineers)에서 1994-1996년 동안 개발되어 표준으로 발간된 "민간 항공 시스템 및 장비의 안전도 평가 수행 지침 및 방법(Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment)"

이와 같이 안전한 소프트웨어를 개발하는 것은 소프트웨어가 가지고 있는 여러 가지 속성 중에서 특히 안전기능이 올바르게 선정되었는지를 확인할 수 있어야 하고, 최종 개발 결과물이 수행하는 안전기능이 정상적으로 작동하는가를 확인하는 것이다. 때문에 앞서 기술한 안전한 소프트웨어 개발의 핵심 내용에 대한 안전 표준의 세부 활동 및 기법/산출물 내에 반영이 가장 큰 차이가 될 것이다. 기존 소프트웨어 품질 관리 프로세스에 안전 프로세스 적용을 위해서는 전체 개발 프로세스적인 측면에서는 위험분석 및 평가 추가, 안전 기능 및 안전 무결성 도출이 추가되어야 하고, 세부적인 내용에 있어 안전 기능이 주어진 안전 무결성 수준을 만족하는지에 대한 각각의 기술, 평가, 방법과 목표 등 전체 생애주기에 걸쳐 관리하여 기능안전성을 확보하기 위한 안전 보증 활동들이 고려되어야 한다.

또한 소프트웨어 안전 프로세스는 시스템 안전 프로세스와 통합되어야 전체 시스템 안전 확보가 가능하다.

[그림 2-8] 소프트웨어 개발 프로세스와 시스템 안전 프로세스와 통합된 소프트웨어 안전 프로세스



출처 : Barbara J. Czerny et al.(2003), A Software Safety Process for Safety-Critical Advanced Automotive Systems, PROCEEDINGS of the 21st INTERNATIONAL SYSTEM SAFETY CONFERENCE

제2절 소프트웨어 안전 프로세스

1. 개요

본 절에서는 산업 분야별 소프트웨어 안전 국제 표준에 대한 이해 및 특징에 대하여 알아보기로 한다. 국제 기능안전 모표준인 IEC 61508 (Functional Safety of Electrical/ Electronic/ Programmable Electronic Safety-related Systems)에 대해 알아보고, 이어서 항공 전자 시스템 분야 소프트웨어 안전 기술 표준인 DO-178C, 철도 신호 시스템의 소프트웨어 신뢰 안전 표준인 IEC 62279, 자동차 전기/전자 시스템의 안전 표준인 ISO 26262 (Automotive Functional Safety Standard)등에 대해서 살펴보기로 한다. 각각의 프로세스의 구성단계는 설명하되, 품질 프로세스와 유사한 활동은 가능한 설명에서 제외한다.

2. 소프트웨어 안전 표준 및 프로세스 특징

1) 기능 안전 모 표준, IEC 61508

(1) 개요

1998년 IEC(International Electrotechnical Commission)에서는 전기, 전자, 프로그램 가능한 전자시스템의 기능안전 표준으로 IEC 61508을 발표하였으며[14], 모든 종류의 산업에 적용 가능한 기본적인 기능 안전 표준으로 작성되었으나, 향 후 각 산업 특성에 맞는 표준들이 개발되었다. 최근 국내외에서 여러 산업에 소프트웨어에 의한 제어가 늘어남에 따라 안전시스템의 관리 방법론으로 IEC 61508 표준에 대한 관심이 급격히 증가하고 있다.

IEC 61508은 안전생명주기, 하드웨어, 소프트웨어 등 세 가지에 대한 안전성 구현 방법 및 검증 방법을 제시하고 있는데, 안전관련 시스템은 IEC 61508에서 정의된 안전수명주기에 따라 위험을 분석한 후 안전 무결성 수준(SIL: Safety Integrity Level)을 설정하고, 하드웨어와 소프트웨어를 목표된 수준(SIL 수준)에 따라 제작한다. IEC 61508은 설치, 운영, 유지보수, 변경, 폐기까지 활동이 포함되어 있는데, IEC 61508의 구성은 아래 표와 같다.

14) 2010년 신규 버전 발표

<표 2-1> IEC 61508의 구성

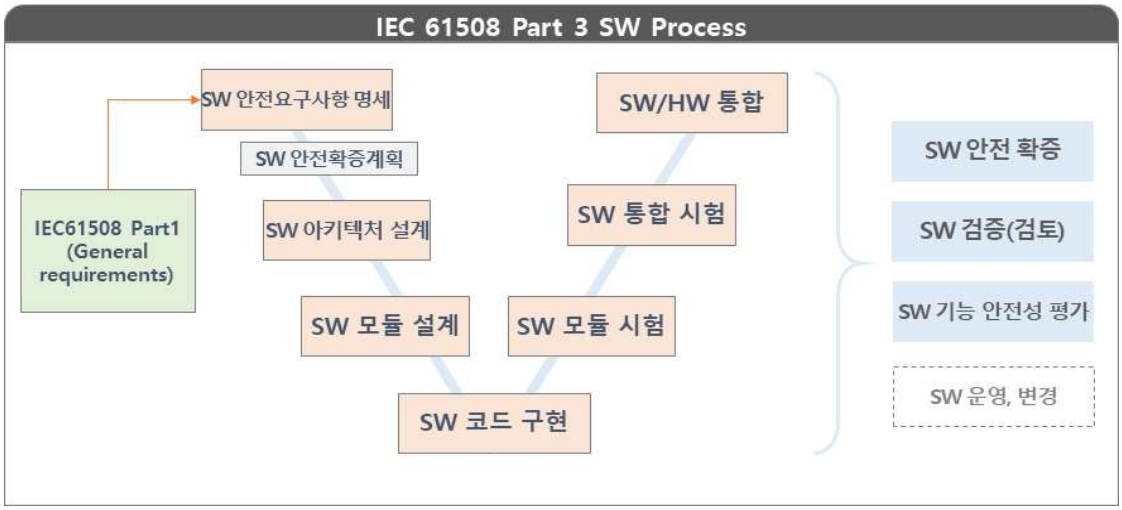
구분	구성
Part 0	기능안전성과 IEC 61508
Part 1	일반 요구사항
Part 2	전자/전자/프로그램 가능한 전자장치 안전 관련 시스템의 요구사항
Part 3	소프트웨어 요구사항
Part 4	정의와 약어
Part 5	안전 무결성 수준 결정 방법의 예
Part 6	IEC 61508의 Part2와 Part3의 적용 지침
Part 7	기법과 수단의 개요

IEC 61508의 Part3에는 안전한 소프트웨어를 개발하는데 요구되는 절차, 활동 및 산출물이 정의되어 있고, 부록에는 소프트웨어의 SIL을 달성하기 위한 기술, 측정이 제시되어 있다.

(2) IEC 61508의 소프트웨어 안전 개발 프로세스

IEC 61508에서는 안전수명주기를 따라 안전 시스템을 개발함으로써 안전 무결성 수준을 달성한다. 전체 소프트웨어 안전개발 수명주기는 각 단계의 목적과 요구사항 및 세부적인 내용이 제시되어 있고 보다 상세한 내용은 IEC12207을 참조하도록 하고 있다.

[그림 2-9] IEC 61508 소프트웨어 프로세스 구성



안전한 소프트웨어 개발을 위해서는 전체 시스템 안전계획에 따라 소프트웨어 안전 기능이 정의되어야 한다. 소프트웨어 개발 프로세스와 유사하게 안전 요구사항 명세, 소프트웨

어 구조 설계, 개발, 시험으로 구성되어 있다. 기능 안전 표준인 IEC 61508의 Part 3에 제시된 소프트웨어 안전 개발 프로세스는 아래와 같다.

소프트웨어 안전요구사항 명세 활동에서는 요구된 안전기능을 구현하기 위해 필요한 각 안전관련 소프트웨어 안전 기능들을 도출하고 소프트웨어 안전 무결성 요구사항을 정의한다.

소프트웨어 구조 설계 단계에서는 하드웨어/소프트웨어상호작용의 중요성을 고려하여 안전 무결성 수준을 충족하는 소프트웨어 구조를 개발한다. 안전 분석 및 검증이 가능하도록 안전 무결성 수준에 따라 상세 설계를 하고 개발한다. 소프트웨어는 안전하게 변경될 수 있도록 설계되어야 한다.

요구되는 소프트웨어 안전기능과 소프트웨어 안전 무결성에 따라서 소프트웨어 안전 요구사항이 달성되었음을 검증하고 의도되지 않은 기능은 수행하지 않음을 증명한다. 소프트웨어와 하드웨어의 통합 시험을 수행하고 의도된 기능대로 올바로 수행하는 것을 증명한다. 각 단계에서 안전 무결성 수준에 따라 적정한 소프트웨어 지원도구를 선택하여 사용한다.

(3) IEC 61508의 소프트웨어 안전 프로세스 특징

가) 위험 분석 및 안전 기능 도출

안전 무결성의 도출 및 평가의 기준으로 IEC 61508은 Safety Integrity Level(SIL)을 명시하고 있다. 이는 안전제어시스템의 안전 무결성으로서, 위험원에 따라 요구되는 수준이 결정되고 이는 다시 정성적 및 정량적인 기법들을 통하여 평가된다.

SIL은 총 4등급으로 구분되고 SIL 1보다 SIL 4가 훨씬 높은 수준의 안전 무결성을 갖는다. IEC 61508은 SIL 평가 척도에 관하여 고장률에 기반을 둔 수식을 제공하고 있다. 따라서 수식을 적용하기 위해서는 표준에서 요구하는 다양한 고장률 산출이 우선되어야 한다. 이를 위하여 고장률 평가 방법인 FMEDA(Failure modes, effects, and diagnostic analysis), FTA(Fault Tree Analysis), ETA(Event Tree Analysis) 등과 같은 분석 기법들이 활용되고 있다.

위험 분석 과정이 포함된 IEC 61508에서 안전 기능 도출 과정은 6단계인데, 시스템 개념 정의, 범위 정의, 위험 분석, 안전 요구사항 도출 및 할당, 안전 기능 상세로 구성되어 있다.

IEC 61508에서는 무결성 수준(Integrity Level)을 만족시키기 위해 위해요인(Hazard)을 분석하여 위험감소 대상을 식별하고 ISO9001과 같이 조직, 프로세스 등이 식별된 위험을 감소시키기 위해 어떤 활동을 해야 하는지 기술하고 있다.

<표 2-2> 위험 분석 및 안전 요구사항 할당

구분		수행할 안전 요구사항
1. 개념		EUC(Equipment Under Control) 이해 필요한 제어 기능 및 물리적 환경 정의 유해 사건 원인 결정, 위험원 정보, 현재 안전규정(법/제도) 등 파악
2. 범위 정의		EUC와 제어시스템의 경계 결정 위험원과 리스크 분석 적용 범위(프로세스, 환경) 결정 고려할 외부 사건, 연관된 장비/시스템, 고려할 사건 유발 유형 결정
3. 위험원 및 위험 분석	위험원 식별	위험원, 위해 사건 식별 (위험원 제거/감소 도 고려) 위험한 상황(결함, 예상 가능한 오용, 악의, 비 허가) 결정
	위험성 계산	결정된 위해 사건으로 이어지는 사건 순서를 결정 명시된 상황에서 위해 사건 발생 가능성 평가 결정된 위해 사건과 연관된 잠재 결과 확인(심각도) -정량적 또는 정성적인 위험원 및 리스크 분석 기법 적용 위험한 사건과 관련된 위험 결정(평가/추정)
4. 안전 요구사항 명세		기능 안전성 달성(위험 제거/감소) 전체 요구사항 명세 개발 안전 기능(Safety Function) 요구사항 명세 안전 무결성(SIL, safety Integrity Level) 요구사항 명세 -목표 안전 무결성 요구사항 - 허용 리스크에 부합 -필요한 리스크 감소 : 허용 가능한 리스크 달성 -허용 가능한 사건 : 허용 가능한 리스크 충족
5. 안전 요구사항 할당		안전 기능, 안전관련 시스템, 위험 감소 수단에 할당 -안전 기능이 허용 가능한 리스크 달성하도록 할당 지속 -달성이 어려우면 명세를 변경하면서 지속적으로 할당 반복 -전체 안전 기능이 할당되고, 목표 고장 기준이 안전 기능에 할당 안전 무결성 수준(SIL), 각 안전 기능에 할당 -확률을 조합하여 적절한 기법 이용, 공통 원인 고장 가능성 고려 -목표 고장 기준 : 저요구/고요구/연속적 작동 모드

출처 : 정보통신산업진흥원(2017), 『공통 분야 소프트웨어신뢰·안전성 확보를 위한 가이드』

나) 안전성 관리

안전성 관리는 사고의 원인이 되는 위험원(Hazard)의 크기를 상대적으로 정량화한 위험성(Risk)이 허용할 수 있는 수준으로 제어된 상태를 의미한다. 안전성 관리를 위한 국제 규격 IEC 61508은 수명주기 각 단계의 안전에 대한 요구사항을 제시하고 있다. 각 단계의 요구사항은 규격을 적용하는 시스템의 특징 및 범위에 따라 구분되지 않고 모두 적용할 수 있도

록 보편화되어 있다. IEC 61508을 적용한 국내외 인증기관들의 입증자료인 위험분석 기법이나 위험원 목록의 양식이 다른 이유가 표준에 상세한 내용이 기술되어 있지 않기 때문이다. 이는 표준의 일반적인 특성이기도 하나, IEC 61508이 모든 산업 시스템에 적용되도록 제정되었기 때문이기도 하다.

안전성 관리는 안전관련 활동 또는 안전수명주기에 대한 책임자가 해야 할 일로 단계별 수행자의 역할, 타 조직과의 인터페이스, 기능안전성 달성 정책 및 수단, 위험한 사건 분석 및 반복 최소화 등을 정의하고 있다. 안전관련 시스템에 관하여 위험원 분석 및 기능 안전성 평가, 각 단계의 산출물 검증, 안전 검증 계획, 위험 사건의 보고 및 분석 등의 활동을 수행한다.

2) 자동차 분야 소프트웨어 안전 표준, ISO 26262

(1) 개요

ISO 26262는 차량 시스템의 잠재적인 재난요인 분석과 위험 평가를 실시하고 이에 따른 안전 목표와 기능 안전 요구사항을 식별하여 관련 위험을 최소화하거나 제거하여 차량의 기능이 용인할 수 없는 위험한 상태를 초래하지 않도록 하는 일련의 안전프로세스이다.

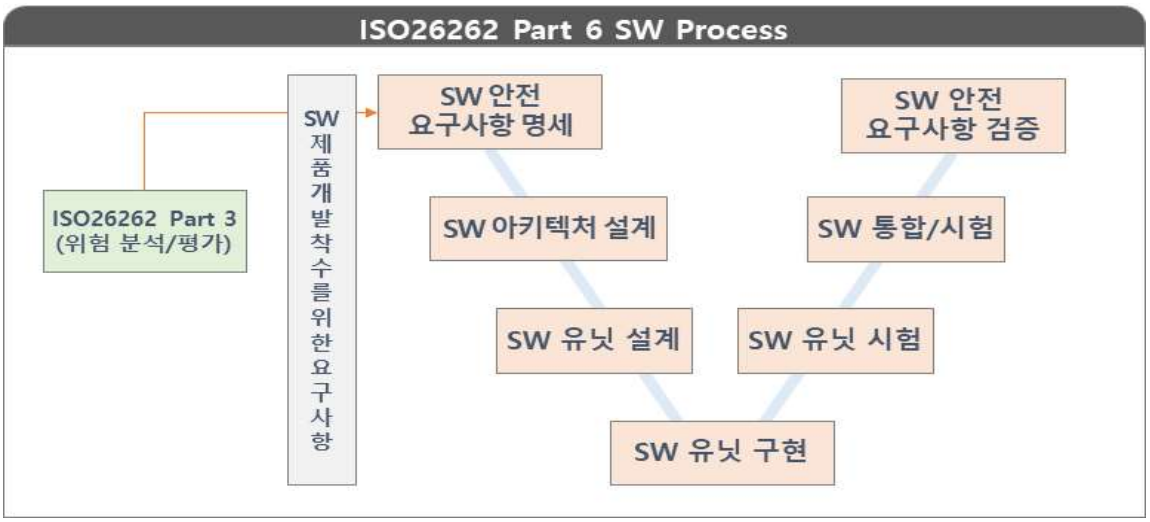
자동차 전기/전자 시스템 안전을 위하여 IEC 61508을 근간으로 제정되었고, 주요 자동차 제조업체에 채택되고 있다. ISO 26262 전에는 안전 관련 자동차 시스템을 위한 소프트웨어의 개발은 주로 자동차 산업 소프트웨어 신뢰성 협회(MISRA, The Motor Industry Software Reliability Association)의 가이드라인이 적용되었다.

IEC 61508은 전기전자장치 안전에 관한 포괄적 규격으로 독일 자동차 업계에서 2000년대 초반 자동차 분야에 적용하였지만 보완이 필요하였다. IEC 61508은 제어 시스템과 안전 메커니즘을 별개로 고려하는데 반해 차량은 기본적으로 이동성을 전제로 하는 시스템으로서 제어 시스템과 안전 메커니즘은 통합되어야 하고 반복적인 개발 생명주기를 가지며, 완성차 업체와 많은 부품공급업체 간의 전문화, 분업화된 생산 방식으로서 이러한 특성이 차이가 있다[15].

15) 출처 : 정보통신산업진흥원, 자동차 소프트웨어 플랫폼 표준 동향

(2) ISO 26262의 소프트웨어 안전 프로세스

[그림 2-10] ISO 26262 소프트웨어 프로세스 구성



소프트웨어 개발 프로세스는 안전요구사항 명세, 아키텍처 설계, 구현, 시험, 통합/시험, 검증의 단계로 구성된다. 각각의 상세한 설명은 ISO/IEC 12207 소프트웨어 개발 프로세스에 설명되어 있으며, 특징적인 부분만 기술한다.

소프트웨어 안전요구사항 명세 단계에서 특징적인 사항은 하드웨어와 소프트웨어간의 인터페이스 요구사항 상세화이다. 소프트웨어 안전 요구사항과 하드웨어와 소프트웨어간의 인터페이스 요구사항이 시스템 안전요구사항과 시스템설계 명세서와 일치함을 검증해야 한다.

소프트웨어 아키텍처 설계 단계는 소프트웨어 안전 요구사항을 구현하는 소프트웨어 아키텍처 설계를 개발, 검증하며, 각 단계를 수행할 때 고려할 점은 소프트웨어 안전 요구사항에 따라야 한다는 점이다. 소프트웨어 단위시험 단계는 요구사항에 정의된 기능을 수행함을 시험하는 것과 동시에 원하지 않는 기능은 포함하지 않는다는 것을 증명해야 한다. 소프트웨어 통합 및 시험 단계, 요구사항 검증단계에서는 임베디드 소프트웨어에 소프트웨어 아키텍처 설계가 구현되고 안전 요구사항을 만족한다는 것을 증명해야 한다.

(3) ISO 26262의 소프트웨어 안전 프로세스 특징

가) 위험 분석 및 안전 무결성

안전 생명주기에서 첫 번째 단계인 개념단계는 주로 완성차 업체가 수행하는 부분으로서

기능안전성 표준의 특징이 잘 나타나있는 단계이다. 여기서는 개발 제품의 범위를 정의한 후 차량수준에서 위험 식별 및 심사를 통해 개발 제품이 안전성에 미치는 영향을 바탕으로 안전성 등급을 설정한다.

위험의 식별은 자동차의 운행상황을 고려하여 개발 제품의 고장으로 인해 나타날 수 있는 위험 사건을 파악한다. 각각의 위험사건은 3가지 요소를 결정함으로써 차량 무결성 등급(ASIL, Automotive SIL)이 결정된다. ISO 26262의 차량 무결성 등급은 위험 상황에 노출 가능성, 위험의 잠재적 심각도 그리고 통제가능성에 따라 결정된다. 차량 무결성 등급은 최저 등급인 ASIL A 부터 최고 등급인 ASIL D 등급 등 총 4개의 등급으로 구분된다. ASIL에 따라 제품의 안전성 목표가 정해지면 이를 통해 세부적인 안전 요구사항을 도출하고 기능 안전 개념을 기술한다.

I나) CMMI, Automotive SPICE 와의 관계

업계에서 일반적으로 품질 인증을 위해 많이 도입하는 것이 CMMI(Capability Maturity Model Integration) 인증이라고 할 수 있다. 그러나 CMMI는 조직의 능력(capability)을 대상으로 심사하고 개선하는 것을 목적으로 하는 것에 비해 ISO 26262는 자동차(혹은 부품) 제품을 대상으로 요구되는 안전성 보장 활동에 대한 심사로서 서로 차이가 있으며, CMMI가 약 2년에 1번씩 자격 갱신이 필요한 것에 반해 ISO 26262는 제품이 대상이므로 유효 기간에 대한 정의가 없다.

근래에 자동차 분야에서 품질향상을 위한 심사모델로서 CMMI와 더불어 유럽을 중심으로 확산되고 있는 것이 Automotive SPICE 이다. Automotive SPICE는 일반 정보기술 분야가 대상인 SPICE를 자동차 분야 특성을 반영한 것이다. ISO 26262에서는 CMMI 및 Automotive SPICE 모두 안전성 등급 ASIL이 해당되지 않을 때 충분한 것으로 구분하고 있으며, 최하위 안전성 등급인 ASIL A가 요구될 때부터 ISO 26262에 정의한 방법을 적용하여 요건을 충족해야 한다.

ISO 26262는 제품의 안전 활동 관리가 기존의 개발 프로세스에서 추가적으로 포함되어 있는 표준이다. 주로 포함되는 활동은 ISO 26262 적용 대상이 되는 아이템 정의, 위험 분석 및 리스크 평가, 안전 요구 사항 분석 및 관리, 안전 분석이 있다. 안전 수준에 따른 안전성 평가 부분을 전기・전자 장치의 개발 프로세스에서 생산을 위한 양산 전에 필수적으로 수행하여 안전성 확보를 강조하고 있다.[16]

16) 우경일 외(2013), ISO 26262 에서 요구하는 안전 활동 관리, 전자공학회지

3) 철도 분야 소프트웨어 안전 표준, IEC 62279

(1) 개요

IEC 62279는 2002년 철도신호시스템의 소프트웨어측면에서 신뢰성과 안전성을 확보하기 위한 기준으로 소프트웨어의 안전성 요건을 규정하고 있다. 소프트웨어에서 필요한 안전요구사항과 관련 기능은 IEC 61508과 IEC 62278에 의해 규정되고 IEC 62279에서는 이러한 요구사항과 기능을 달성하고자 하는 수단들을 제시한다. 이 규격의 핵심은 소프트웨어 안전 무결성 등급(Safety Integrity Level : SIL)에 있다. 안전 등급을 0에서 4까지 5단계로 구분해서 0등급은 안전과 관련이 없는 수준이며 1에서 4등급으로 갈수록 안전에 치명적인 영향을 미친다는 것을 의미한다.

이 규격에서 설명하고 있는 소프트웨어의 개발 프로세스는 소프트웨어의 요구사항, 구조, 설계 및 구현, 증명 및 시험, 소프트웨어와 하드웨어의 통합, 검증, 평가, 유지보수, 확인, 품질보증의 절차로 나누어져 있고 각 절차마다 수행할 업무와 생산할 항목들에 대해서도 정의하고 있다.

철도는 미국은 화물 중심, 유럽은 승객 중심으로 발달하고 있어, 안전성 관련해서는 유럽의 표준이 더 엄격히 적용되고 있다. 철도 표준은 철도 시스템과 부품 사양, 개발 및 운용의 각 프로세스에 대해 안전 요구사항을 규정하는 European Norm(EN)계열의 안전성 표준들이 존재하며 철도 시스템의 안전성을 확보하기 위한 RAMS[17] 활동에 대한 표준으로 구성되어 있다,. 철도 시스템에 관련된 안전 표준에는 IEC 62278 (RAMS[18], EN 50126), IEC 62279 (소프트웨어, EN 50128), IEC 62425 (EN 50129, 신호), IEC 62280 (EN 50159, 통신)이 있다.[19]

네 가지 표준은 일반산업 분야 전기전자 안전 표준인 IEC 61508, UIC(French: Union Internationale des Chemins de fer or International Union of Railways : 국제철도연합)의 기술지침과 유럽 각국의 철도 시스템의 기술요구사항을 통합하였다. IEC 61508의 안전성 분석 생명주기 모든 단계, 위험분석 후 결정하는 안전 무결성 등급(Safety integrity level, SIL)을 도입하였다. [20]

---

17) 시스템의 신뢰성(Reliability), 가용성(Availability), 유지보수성(Maintainalility), 안전성(Safety)의 약자로, 철도와 같이 사소한 고장발생이 열차지연 또는 중대사고로 발전되는 등 안전이 필수적인 분야에서 그 중요성이 강조되고 있으며, '철도안전법'에서는 철도운영기관이 안전을 확인할 의무를 명시하고 있음.

18) Railway application-specification of demonstration of reliability, availability, maintainability and safety

19) SPRi(2015), 소프트웨어 안전 산업동향 조사, pp24, IEC 62279 (철도 응용 분야 - 통신, 신호 및 처리 시스템 - 철도 제어 및 보호 시스템 용 소프트웨어) 및 IEC 62425 (철도 응용 분야 - 통신, 신호 및 처리 시스템 - 신호 관련 안전 관련 전자 시스템)

(2) IEC 62279의 소프트웨어 안전 프로세스

IEC 62279 : 2015는 철도 제어 및 보호 응용 프로그램을 위한 안전 관련 소프트웨어의 개발, 배포 및 유지 관리 시 준수해야 하는 일련의 요구 사항을 제공한다. 또한, SIL 수준에 따른 조직 구조, 조직 간의 관계, 개발, 배포 및 유지 관리 활동과 관련된 책임에 대한 요구 사항도 정의하고 있으며, 관련 인원의 자격 및 전문성에 대한 기준도 이 기준에서 제공한다. IEC 62279의 구성은 아래 표와 같다.

<표 2-4> IEC62279의 구성

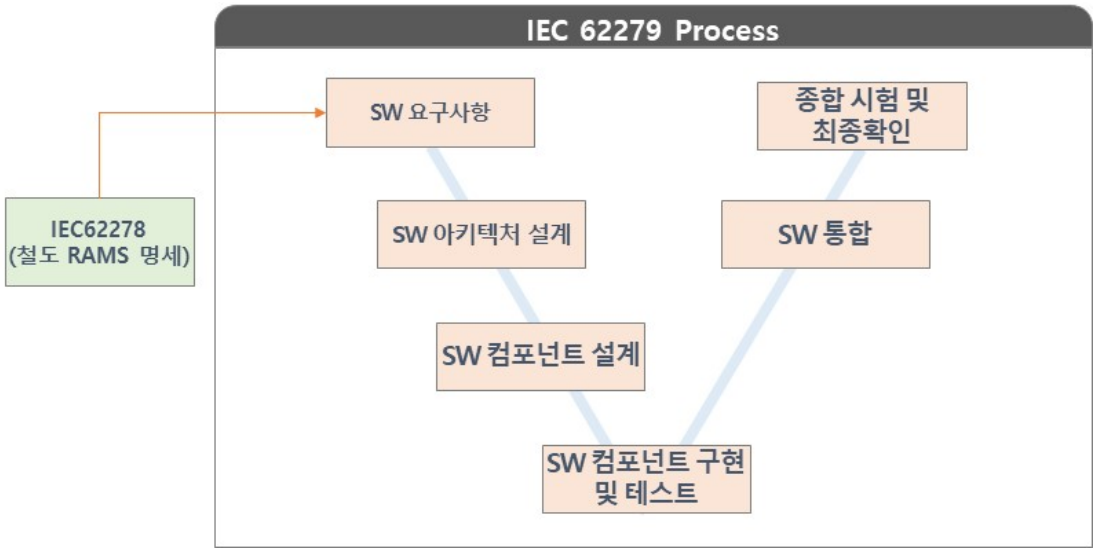
장	내용	장	내용
1	범위	8	응용 프로그램 데이터 또는 알고리즘 개발
2	관련 표준	9	소프트웨어 배포 및 유지 관리
3	정의와 약어	Annex A	기술 및 대책 선택 기준
4	목표, 적합성 및 소프트웨어 안전 무결성 레벨	Annex B	주요 소프트웨어 R & R
5	소프트웨어 관리 및 조직	Annex C	문서 통제 요약
6	소프트웨어 보증	Annex D	기술 목표 및 설명
7	일반적인 소프트웨어 개발	-	

IEC 62279는 철도 제어 및 보호 어플리케이션에 사용되는 프로그램 가능한 전자 시스템용 소프트웨어 개발을 위한 프로세스 및 기술 요구 사항을 규정하며, 안전에 영향을 미치는 모든 영역에서 사용된다. 이러한 소프트웨어에는 응용 프로그램 프로그래밍, 운영 체제, 지원 도구, 펌웨어 등을 포함하며, 응용 프로그래밍은 고급 프로그래밍, 저 레벨 프로그래밍 및 특수 목적 프로그래밍에도 적용한다.

IEC 62279 소프트웨어 개발 프로세스는 SW요구사항, SW 아키텍처 설계, SW 컴포넌트 설계, 구현 및 테스트, SW통합, 종합 시험으로 이루어져 있으며, 각각의 단계는 일반 소프트웨어 개발프로세스와 유사하나, 각각의 단계에서 위험 분석 및 안전 설계, 개발, 시험 단계가 포함된다.

20) NIPA(2017), SW안전가이드 철도 분야

[그림 2-11] IEC 62279 소프트웨어 개발 프로세스 구성



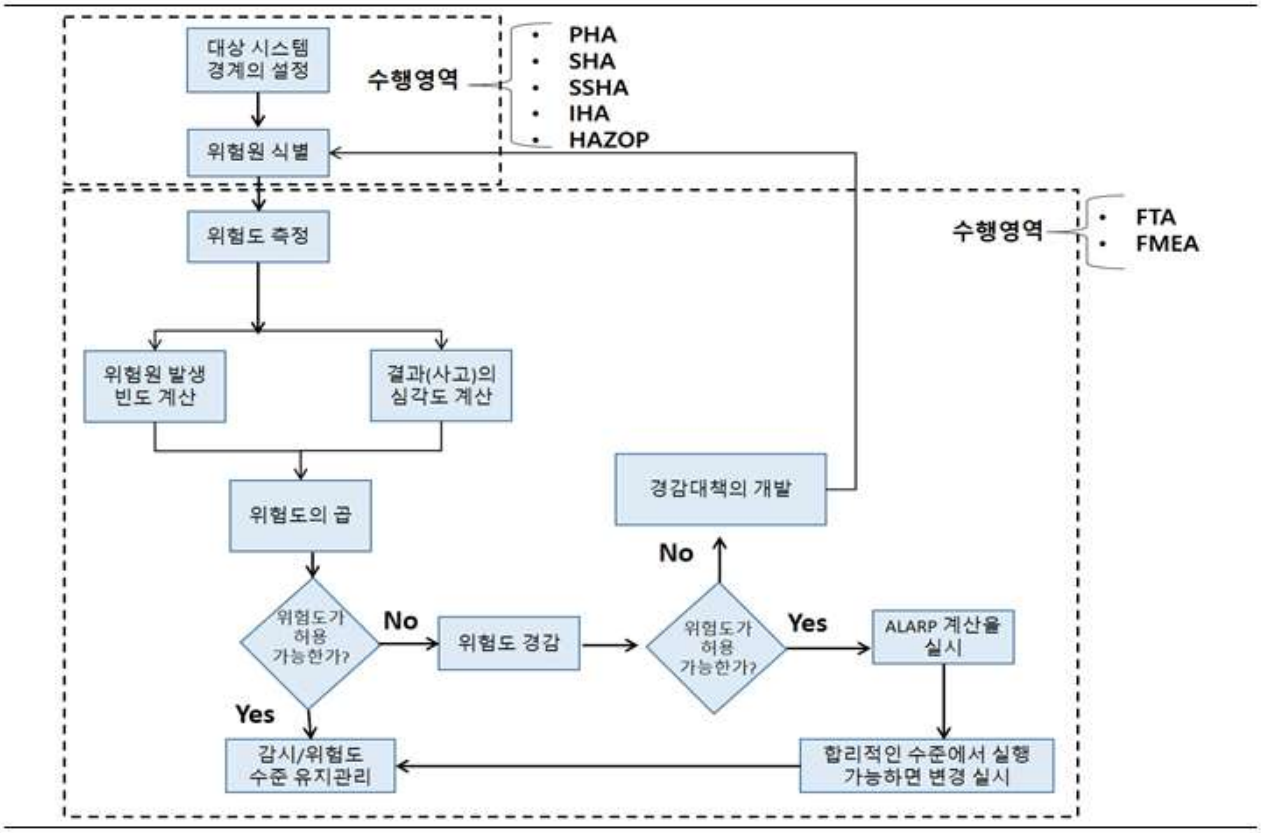
앞 절에서도 언급했듯이, 소프트웨어에 할당된 안전 기능을 지정하는 프로세스를 정의하는 방법은 IEC 62278에 기술되어 있으며, 안전 기능 지정을 위해 다음과 같은 체계적인 접근 방법을 요구하고 있다.

a) 위험을 식별하고, 위험을 평가하고, b) 위험 수용 기준을 충족시키는데 필요한 위험 감소 및 대책을 식별하고, c) 요구되는 위험 감소를 위한 안전장치에 대한 전반적인 시스템 안전 요구 사항 규격을 정의하고, d) 적절한 시스템 아키텍처 선택, e) 안전 요구 사항 명세를 안전 관련 시스템으로 변환하는 데 필요한 기술 및 경영 활동 계획, 모니터링 및 제어 등이 포함되어 있다.

(3) IEC 62279의 안전성 분석

IEC 61508 기반 안전성 분석 시스템 생명주기는 개념 정의, 개발, 제품화, 운영, 폐기의 5 단계로 구성되어 있으며, 대부분의 안전성 분석 수행은 시스템 설계 초기에 수행함으로써 안전성을 확보하는데 주력하고 있다. 특히 개발 단계에서 세부적으로 설계와 시험 단계를 통해 많은 안전성 분석을 지속적으로 수행하고 각 단계마다 수행되는 안전성 분석에 대한 검토가 이뤄진다. 검토를 통해 해당 단계 수준에서 확보해야할 안전성에 대한 분석이 제대로 되었는지 확인한다.

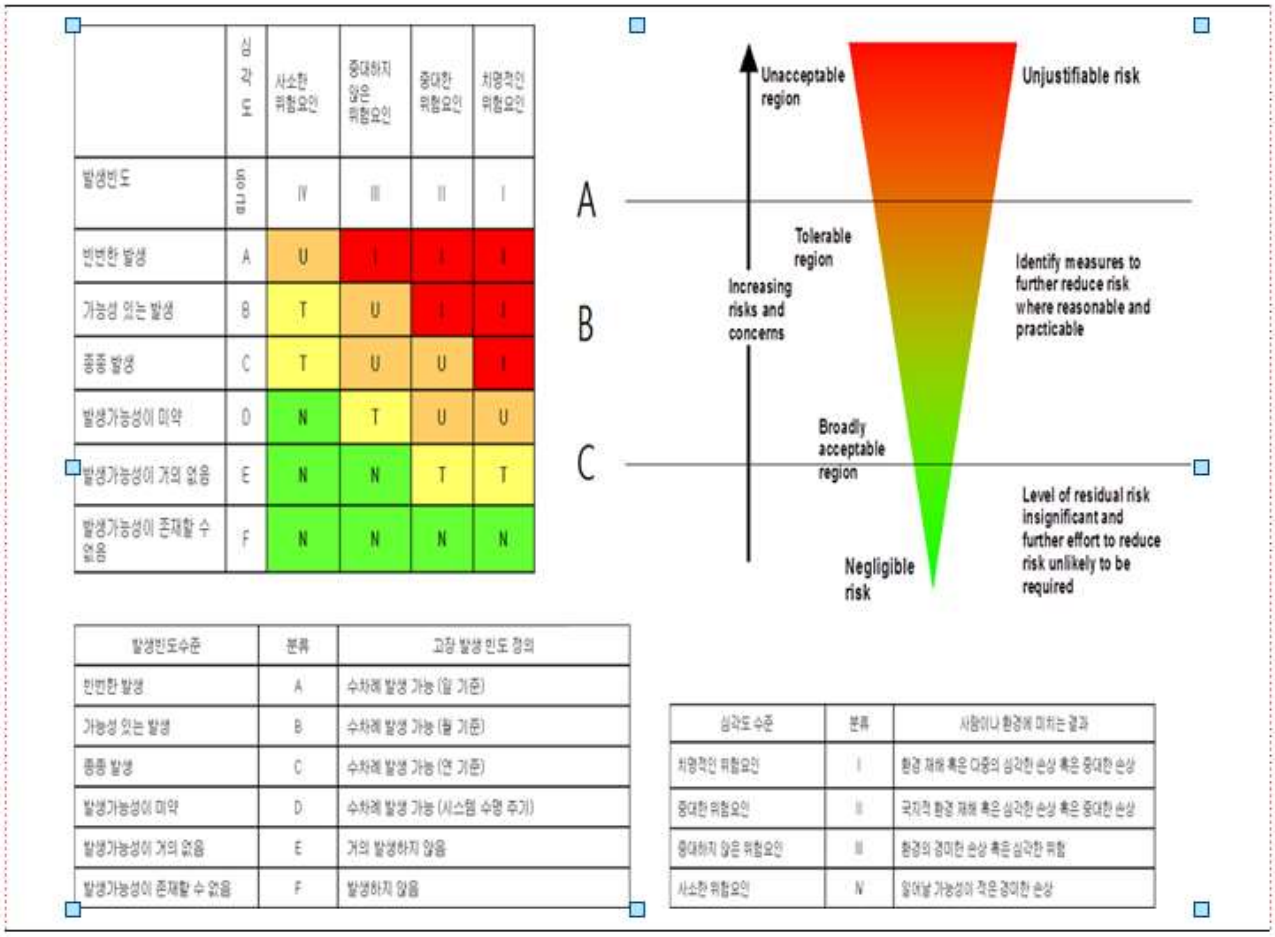
[그림 2-12] 위험도 평가 절차



자료: 정보통신산업진흥원 (2017),『철도 분야(IEC62279) 소프트웨어신뢰·안전성 확보를 위한 가이드』

안전성 분석은 위험도 분석이라고 할 수 있다. 위험도 평가 절차는 크게 두 가지로 구분된다. 위의 점선 사각형 부분은 위험원 분석 활동으로 위험원 정의 및 분석 수행에 해당되는 영역이다. 아래 점선 사각형 부분은 분석 및 정의 과정을 거쳐 추출된 위험요소 및 위험원인을 찾아 영향 정도를 평가하는 단계이다.

[그림 2-13] 위험도 매트릭스와 ALARP



	심각도	사소한 위험요인	중대하지 않은 위험요인	중대한 위험요인	치명적인 위험요인
발생빈도	등급	IV	III	II	I
빈번한 발생	A	U	I	I	I
가능성 있는 발생	B	T	U	I	I
종종 발생	C	T	U	U	I
발생가능성이 미약	D	N	T	U	U
발생가능성이 거의 없음	E	N	N	T	T
발생가능성이 존재할 수 없음	F	N	N	N	N

발생빈도수준	분류	고장 발생 빈도 정의
빈번한 발생	A	수차례 발생 가능 (일 기준)
가능성 있는 발생	B	수차례 발생 가능 (월 기준)
종종 발생	C	수차례 발생 가능 (년 기준)
발생가능성이 미약	D	수차례 발생 가능 (시스템 수명 주기)
발생가능성이 거의 없음	E	거의 발생하지 않음
발생가능성이 존재할 수 없음	F	발생하지 않음

심각도 수준	분류	사람이나 환경에 미치는 결과
치명적인 위험요인	I	환경 재해 혹은 다중의 심각한 손상 혹은 중대한 손상
중대한 위험요인	II	국지적 환경 재해 혹은 심각한 손상 혹은 중대한 손상
중대하지 않은 위험요인	III	환경의 경미한 손상 혹은 심각한 위협
사소한 위험요인	IV	일어날 가능성이 적은 경미한 손상

자료: 정보통신산업진흥원 (2017),『철도 분야(IEC62279) 소프트웨어신뢰·안전성 확보를 위한 가이드』

I : Intolerable(허용 불가능한), U : Undesirable(바람직하지 않은) , T : Tolerable(허용가능한), N : Negligible(무시할 수 있는)

위험도 평가 절차 과정을 거쳐 위험원에 대해서 심각도와 발생빈도에 따라 ALARP(As Low As Reasonably Practicable) 계산을 수행한다. 심각도와 발생빈도 판단을 위한 기준을 제공하며, 두 판단요소에 따라 ALARP를 결정하고, 위험도 평가를 수행한다.

4) 항공기 분야 소프트웨어 안전 표준, DO-178C

(1) 개요

DO-178 표준은 항공전자시스템에서 사용되는 안전 필수 소프트웨어의 안전성을 취급하는 표준이다. FAA(Federal Aviation Administration)가 인증을 위한 기술 표준 오더(Technical Standard Order, TSO)에서, 소프트웨어가 항공 환경에서 신뢰할 수 있게 수행할 수 있는지를 확인하기 위한 표준으로 DO-178C를 사용한다.[21]

항공기 시스템에서 항공소프트웨어는 1960년부터 사용되기 시작하였으며, 현대 무기체계의 경우는 첨단 전자장비 등의 비중이 높아짐에 따라 소프트웨어의 비중과 역할도 비약적으로 높아지고 있다. 전투기가 수행하는 임무 중 항공소프트웨어가 차지하는 비중은 1960년에 F-4 전투기는 8%에 불과했지만, F-16에서는 45%, F-22는 80%, F-35는 90% (2007년)에 달해 F-4에 비해 11배가 증가되었다[22]. 이에 따라 하드웨어 위주의 결함만 해결하던 과거와는 달리 이제는 소프트웨어 오류에 대한 문제해결 및 품질보증에 대한 노력이 절실하게 되었다.

1980년에 이러한 시대적인 요구에 의해 RTCA(Radio Technical Commission for Avionics)는 항공소프트웨어 안전에 대한 검증과 인증을 위한 기준인 DO-178을 제정하였다. 유럽에서도 EUROCAE(European Organization for Civil Aviation Equipment)에 의해 ED-12가 발표되었다. DO-178과 ED-12는 관장하는 기관과 적용 대상만 다를 뿐 사실상 같은 표준이다. 이 두 문서는 여러 번의 개정을 거쳐 1992년에 DO-178B(ED-12B), 그리고 2011년에는 DO-178C(ED-12C)가 승인되었다.

DO-178B와 비교하여 DO-178C에는 모델-기반 개발 및 검증(Model-Based Development and Verification), 정형기법(Formal Method) 그리고 객체지향 기술(Object-Oriented Technology)과 같은 새로운 소프트웨어 개발 기술에 대한 지침이 추가 반영되었다. 새로운 기술을 도입하기 위해 기존의 문서를 확장하기보다는 DO-178C와 함께 부가적으로 사용될 수 있는 새로운 문서들을 추가하였다. 이 문서들을 세부적으로 살펴보면 DO-330(소프트웨어 툴 검증 고려사항), DO-331(모델 기반 개발 및 검증), DO-332(객체지향 기술 및 관련 기술), DO-333(정형 기법), DO-248C(추가 정보) 등이다.

---

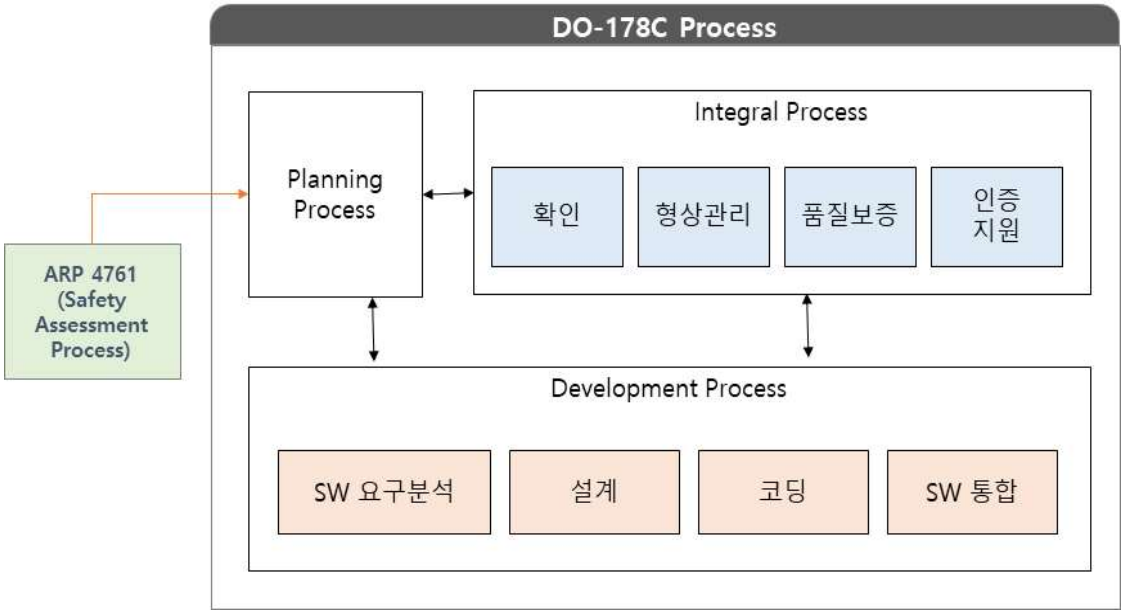
21) SPRi(2015), 소프트웨어안전 산업동향 조사

22) 출처 : http://www.edaily.co.kr/news/read?newsId=01161126619113472&mediaCodeNo=257

(2) DO-178C의 소프트웨어 안전 프로세스

DO-178C에서 말하는 소프트웨어 생명주기 프로세스는 아래 그림과 같이 구성되어 있다.

[그림 2-14] DO-178C 프로세스 구성



각 프로세스에 대한 상세한 내용은 다음과 같다.

가) 소프트웨어 계획(Planning) 프로세스

프로젝트의 소프트웨어 개발 및 통합 프로세스의 활동을 정의하고 조정하는 프로세스이다. 이 프로세스의 목표는 요구 사항을 충족시키고 소프트웨어 수준과 일치하는 신뢰 수준을 제공하는 소프트웨어를 만드는 방법을 정의하는 것이다. 소프트웨어 계획 프로세스는 인증 양상, 개발, 검증, 형상 관리, 품질 보증 계획의 총 다섯 가지 계획으로 구성된다.[23] 인증양상 계획은 소프트웨어 안전성 수준 검증을 위해 활용되는 소프트웨어 생명주기와 데이터, 검사일정, 인증 고려사항 등이 포함된다.

나) 소프트웨어 개발(Development) 프로세스

소프트웨어 요구 사항 분석, 소프트웨어 설계, 소프트웨어 코딩, 소프트웨어 통합과 추적성 프로세스로 구성된다. 소프트웨어 요구 사항 프로세스란 시스템 생명 주기 프로세스의

23) 김희성(2015), 항공용 소프트웨어 인증을 위한 DO-178C 적용절차 개관, SBAS 정책・기술동향

산출물을 사용하여 상위 수준 요구사항을 개발하는 것이며, 기능, 성능, 인터페이스 및 안전 관련 요구 사항이 포함되어 있다. 소프트웨어 설계 프로세스란 상위 수준 요구 사항을 세분화하여 소스 코드를 구현하는 데 사용할 수 있는 소프트웨어 구조 및 상세 요구 사항을 개발하는 것이다. 소프트웨어 통합 프로세스에서는 통합 시스템 또는 장비를 개발하기 위해 소스 코드, 컴파일링 데이터, 링킹 데이터, 로딩 데이터를 통합한다.

다) 소프트웨어 통합(Integral) 프로세스

소프트웨어 생명주기 프로세스 및 출력에 대한 정확성, 제어, 신뢰성을 보장하는 필수적인 프로세스이다. 소프트웨어 확인 프로세스, 소프트웨어 형상 관리 프로세스, 소프트웨어 품질 보증 프로세스 및 인증 (certification liaison) 프로세스로 구성된다. 소프트웨어 확인 프로세스에서는 소프트웨어 개발 중에 나타날 수 있는 오류를 발견하고 보고한다. 안전 관련 형상 관리 중 가장 중요한 사항은 추적성 관련 사항이다. 인증 프로세스는 인증기관과 소통과 이해를 지원하는 것이 목적이다.

DO-178은 소프트웨어 개발프로세스의 특징은 개발프로세스와 별도로 검증 프로세스를 정의했다는 것이다. 소프트웨어개발 V모델과 유사한 흐름을 가지며, 소프트웨어 등급별로 적용해야하는 프로세스를 명시하고 있다. 안전 무결성 등급에 따라 하위요구사항과 소프트웨어 아키텍처가 상위요구사항과 일치하는지는 높은 등급의 소프트웨어를 개발할 때는 검증하도록 되어 있다. 이중에서도 중요한 항목들은 형상관리 시 별도의 베이스라인을 가지고 관리하여 프로세스 수행에 대한 자료를 엄격하게 저장하고 수정하도록 되어 있다.

(3) DO-178C의 소프트웨어 안전 프로세스 특징

가) 안전성 평가

안전성 평가(Safety Assessment)는 시스템 개발의 한 부분으로써 이를 통해 알려진 안전성 요구사항 및 인증 요구사항에 부합하도록 시스템 아키텍처가 설계되고 설치되었는지를 검증한다. 항공전자 산업에서 수행되는 안전성 평가는 SAE (Society of Automotive Engineers, Inc.)에서 제시하고 있는 ARP 4761(Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment)과 ARP 4754(Guidelines for Development of Civil Aircraft And Systems)에 명시된 절차를 따르며, 안전성 요구사항을 식

별하고 이를 검증하게 된다. 항공전자시스템의 설계에 있어 고장 발생 원인을 완전히 제거하는 것이 이상적이지만 설계의 제약조건 등의 문제로 인하여 불가능할 경우는 고장발생 확률 및 시스템의 심각도 등급을 조정하여야 하며 허용수준 이하로 낮추기 위한 설계, 위험성을 최소화하기 위한 안전장치의 추가를 통해 위험성을 낮추는 설계방안을 검토하게 된다. 이러한 방안 외의 경우에는 경보 신호 및 경보장치를 적용하여 운용상의 오작동 및 고장에 대해 추가적으로 반영한다.

시스템을 위험등급에 따라 5가지로 구분하여 개발을 하도록 정의하고 있다. 이 구분은 다음과 같다. 레벨 A부터 D까지는 소프트웨어의 오작동이 발생했을 때 유발할 수 있는 장애의 등급을 나누어서 소프트웨어 등급을 정한 것을 알 수 있다. 레벨 E는 항공기의 운항이나 조종사의 임무에 영향을 미치지 않는 소프트웨어로 본 표준의 대상이 아님을 명시하고 있다. 항공소프트웨어 개발과정에 있어 안전에 대한 검증을 위해서는 위험요소를 결정하는 것이 중요하다. DO-178은 항공기의 안전을 저해하는 결함과 관련된 항공소프트웨어 위험요소를 다섯 가지 등급으로 분류하고 있다. 다섯 가지 등급은 결함이 가장 치명적인 A 등급부터 항공기 안전에 전혀 영향이 없는 E 등급까지이다. A 등급은 항공기에 재난을 초래하는(Catastrophic) 결함을 유발시키는 수준이며, B 등급은 항공기에 매우 위험한(Hazardous) 결함을 초래하고, C 등급은 중대한(Major) 결함을 유발하며, D 등급은 사소한(Minor) 결함을 발생시키는 상태, 그리고 E 등급은 항공기에 아무런 영향이 없는(No Effect) 상태를 나타낸다.

〈표 2-3〉 항공 분야 위험요소 등급[24)]

범주	기술
Catastrophic	소프트웨어의 치명적인 오류로 인해 안전 비행과 착륙을 저해, 추락의 위험을 갖는 고장 위험 수준
Hazardous	비정상적인 동작으로 위해하고 심각한 장애 상황을 야기, 물리적인 변형 또는 과도한 업무 부하로 인해 조종사가 비행을 유지할 수 없거나 심각한 승객의 부상의 위험 수준
Major	비정상적인 동작으로 주요한 안전성을 감소시키는 주요한 장애를 야기하여 사용자의 불편이나 부상 등을 초래함
Minor	비정상적인 동작이 작은 장애를 야기, 항공기의 안전성을 크게 감소시키기 않고 승무원의 활동의 손상되지 않으나 다소 불편함 (비행 경로 변경 등)
No Safety Effect	비행 조정 또는 승무원의 과부하에 영향을 미치지 않는 고장 수준의 소프트웨어

24) Spri(2015), 국내 소프트웨어 안전 산업동향 조사, 위키피디아 편집

또한 DO-178 인증기준을 통과하기 위해서 달성해야 하는 속성들이 정의되어 있는데 이를 충족목표(Objectives)라고 한다. 이 충족목표들은 개발 대상 소프트웨어의 등급에 따라 달라진다. DO-178C에서 A 등급의 경우 71개의 충족목표를 달성해야하는 반면, D 등급의 경우 26개의 충족 목표만을 달성하면 된다.

나) 안전성 관리 및 인증 적합성 평가

DO-178은 5가지 위험성 레벨 외에 일관성과 결정성, 추적성, 독립성, 경로 테스팅, 입증된 도구의 사용 등을 권고하며, 특히 상세한 계획을 매우 중요시한다. 안전 관리 계획은 반드시 개발 활동 전에 완료되어야 하고 모든 내용을 기술해야 하며, 계획 이행에 대한 검증을 포함해야 한다. 수립된 안전 계획은 품질관리자와 함께 작성한 후 점검하고 자격이 있는 DER(Designated Engineering Representative)로부터 승인 받아야 한다. 인증을 위해서는 조직 내에 독립된 품질 담당자가 있어야 하며 여기서 “독립된” 이란 의미는 개발 조직에 포함되지 않은 사람을 말한다. 또한 팀 내에는 프로젝트 관리자와 함께 필수적으로 시험 엔지니어와 형상관리 담당자가 개별적인 역할자로 존재해야 한다.

계획 및 개발 작업에 참여하여 DER이 제품 생산에 대하여 확인하고 관여하는 인증 활동을 SOI(Stage Of Involvement)라고 한다. 각 단계는 계획 수립 단계에 관한 확인인 SOI #1, 개발 단계에 관한 확인인 SOI #2, 검증 및 시험 단계에 관한 확인인 SOI #3, 마지막 인증 적합성을 확인하는 단계인 SOI #4 로 구성되어 있다. 각 확인 단계마다 DER의 검토가 진행된다. 특히 SOI #3가 전체 인증의 고비인데, 이 단계에서 시험이 완료될 필요는 없지만, 시험이 계획대로 진행된다는 것을 충분히 보여줘야 한다.

제3절 소프트웨어 안전 미적용 사고 사례 및 표준 적용 사례

본 절에서는 안전 미적용 또는 일부 누락으로 인한 사고 사례를 찾아 소프트웨어 안전 미적용에 대한 실패 사례로서 살펴본다. 성공 사례는 영국의 IEC 61508 협회 및 CASS Scheme을 살펴보고, 국내 사례로는 산업안전보건원의 안전 적용 사례를 살펴본다.

1. 소프트웨어 안전개발 프로세스 부재로 인한 사고 사례

안전이 필수적으로 요구되는 시스템에서 소프트웨어의 신뢰성 및 안전성 문제에 의한 사고들이 많이 존재한다. 이 사고들의 공통점은 한 번의 사고로 대규모의 재산과 인명 피해를 낼 수 있다는 점이다. 다음은 소프트웨어의 신뢰·안전성이 보장되지 못함에 의한 사고 사례들이다. 소프트웨어 버그로 인해 발생한 사고 중 최초로 발생되어 안전 표준 제정의 발단이 되었던 사건이 Therac 25 사고이다.

1) 사고 경위

1985년에 캐나다의 AECL[25]이란 회사는 암 종양 제거를 위한 방사선 치료기인 Therac 25라는 제품을 판매하기 시작했다. Therac 25는 전자빔으로 피부 근처의 종양을 제거하는 전자(Electron)모드와 엑스레이(X-ray)로 피부 깊숙한 곳의 종양을 제거하는 엑스레이(X-ray)모드의 두 가지 동작 모드를 지원했다.

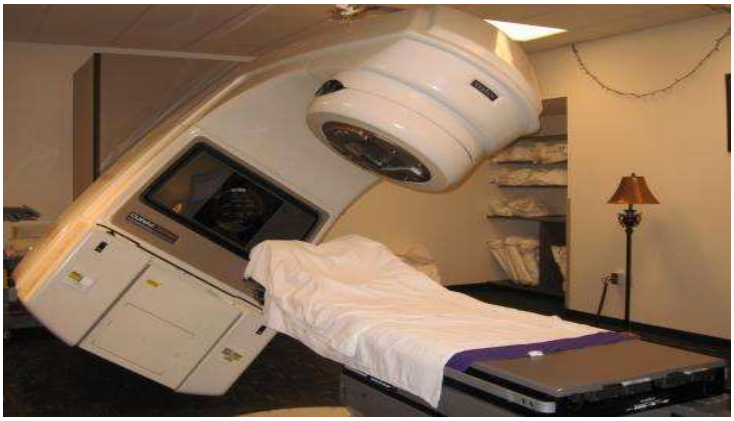
Therac 25는 두개의 장비로 되어 있던 것을 하나로 합쳐 공간과 비용을 절약하도록 개선한 장비이었다. Therac 25 장비는 전자모드와 엑스레이모드가 있다. 엑스레이모드는 턴테이블이라고 하는 장치를 장비와 시술부위 중간에 위치시켜 강한 방사선을 안전하게 환자에 쪼일 수 있도록 조절되어야 한다. 턴테이블의 제어는 소프트웨어를 이용하는데, 소프트웨어 버그로 인해 간혹 엑스레이모드인 상태에서 턴테이블이 제 위치에 있지 않은 경우가 발생했다. [26]

---

25) Atomic Energy of Canada Limited, 1952년 설립, 캐나다형 원전 개발, 판매 등 상업적 활동 및 원자력 연구개발, 폐기물 관리 및 동위원소 생산과 같은 공공정책적 역할 수행. 오늘날 AECL은 원자력 기술부터 물리학, 야금학, 화학, 생물학 및 공학 분야의 전문 지식을 통한 응용 프로그램 개발

26)출처:https://m.blog.naver.com/PostView.nhn?blogId=saiparan&logNo=90080535747&proxyReferer=&proxyReferer=https%3A%2F%2Fwww.google.co.kr%2F

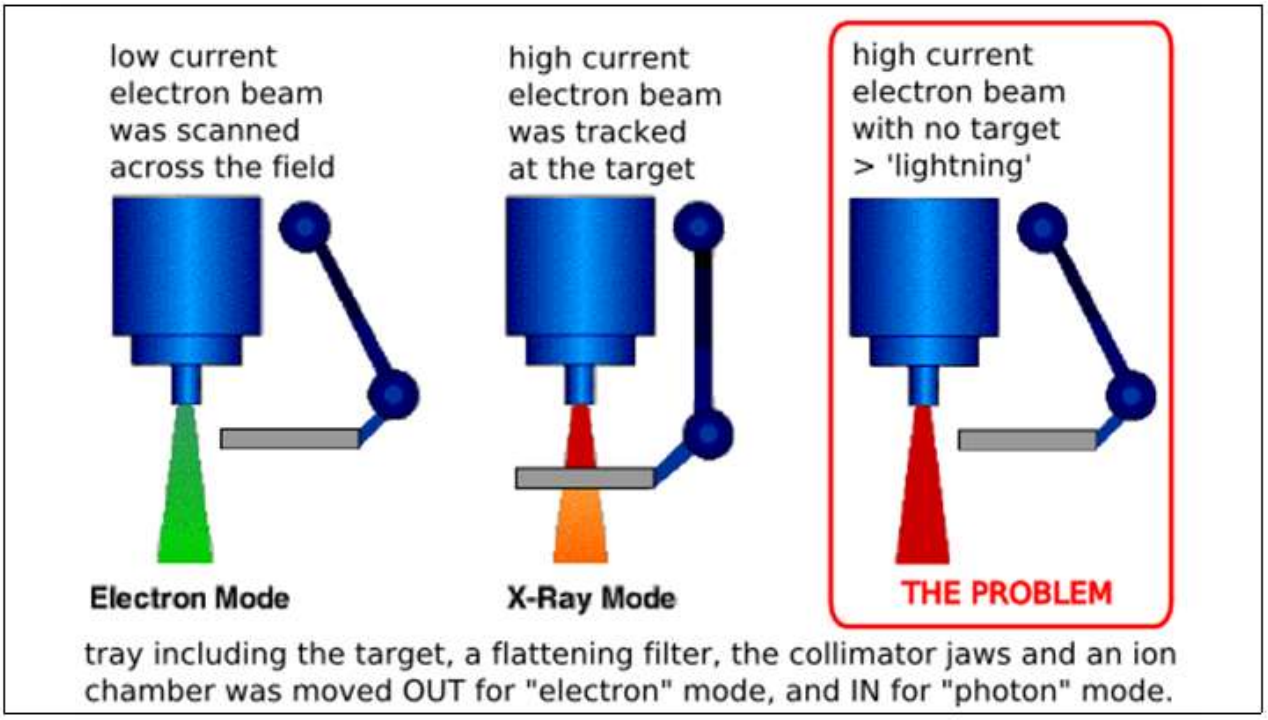
[그림 2-15] Therac 25 사진



출처 : http://idg.bg/test/cwd/2008/7/14/21367-radiation_therapy.JPG

정상적인 경우라면 왼쪽의 두 그림중 하나의 상태여야 하는데 기본적인 소프트웨어 오류 처리를 제대로 지키지 않아 오른쪽과 같은 상황에서 기계가 작동하였다. 이로 인해 무려 6건의 사고가 발생해서 3명이 죽고 다른 3명은 심각한 방사능 후유장애에 시달려야 했다.

[그림 2-16] Therac 25 정상작동 및 오작동 설명



출처 : http://radonc.wdfiles.com/local--files/radiation-accident-therac25/Therac25.png

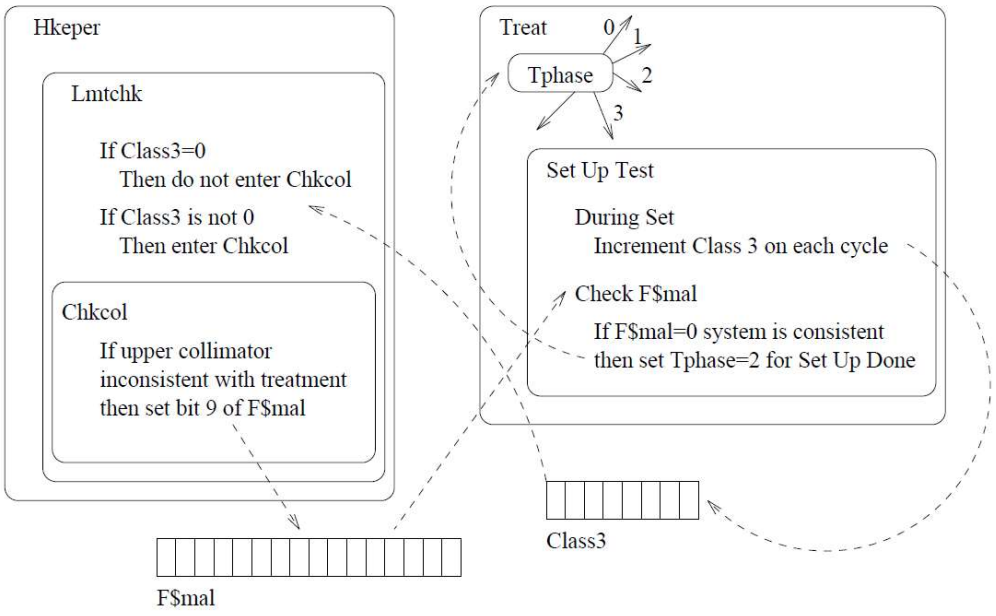
2) 사고 원인

이 사고의 원인으로는 ① 소프트웨어 오류, ② 사용자 인터페이스 및 매뉴얼 부족, ③ 제조기업의 무지 등을 들 수 있다.

① 소프트웨어 오류

여러 종류의 소프트웨어 오류가 있으나, 그 중 한 가지를 소개하겠다. Therac-25의 시술 준비여부를 점검하기 위한 변수인 Class3 변수가 있다. Class3가 0인 경우는 기계가 준비가 되지 않아 시동될 수 없으며, 0이 아닌 경우는 준비 상태이며, 기계 시동을 위한 다음 점검 프로세스(Chkcol)로 넘어가서 기계 시동여부를 검사한다. Therac-25가 시동된 후에는 다시 준비상태로 넘어가며, Class3을 증가하게 된다.

[그림 2-17] 소프트웨어 에러 로직



출처 : N.G. Leveson, C.S. Turner(1993), "Medical devices The Therac25", IEEE

그런데 기계 점검상태에서 Class3 변수를 하나씩 증가하여, 0이 아닌 상태를 만들게 소스코드가 만들어져 있었는데, Class3 변수가 8bit이였고, 256번째 준비상태는 Class3이 0이 되는 문제점이 있었다. 이 문제의 해결은 간단했다. 준비 상태는 Class3을 0이 아닌 일정한 값으로 변환하면 되는 것이다. 이 문제는 제3자에 의한 코드 리뷰만 실시했어도 쉽게 해결할

수 있었던 것이나 소프트웨어에 대한 기본적인 테스트도 실시하지 않았던 시기에 일어난 사고였고 이로 인하여 의료분야에서는 개발 및 점검 프로세스를 강조하는 계기가 되었다.

② 사용자 인터페이스 및 매뉴얼 부족

Therac 25는 기기실에서 기기를 수동으로 조정하는 구조였다. 사용자가 제어실에서 처방 값을 모두 입력하는 과정을 개선하기 위해 DEFAULT 기능을 추가하여 이전 값을 자동으로 입력하는 편의 기능을 추가하게 되었다. 하지만 이러한 구성으로 인해 방사선사들은 소프트웨어의 오류 메시지의 출력을 늦게 발견하였으며 오류 메시지 (MALFUNCTION 54)를 해석하지 못한 상태로 기기를 계속 구동하였다.

사고 장비에 제공되는 기기-운영자 설명서에는 오류의 종류만 나열되고 메시지에 대한 설명이 나타나 있지 않았다. 오류 메시지 중에서는 환자에게 위험할 수 있다는 메시지도 포함만 되어있을 뿐 따로 분류되지 않은 상태였기 때문에 방사선사들은 “MALFUNCTION 54“라는 메시지를 무시하게 되었다. 개발자만 알 수 있는 암호같은 메시지가 불러낸 문제로 이에 대한 정확한 메시지만 제시했어도 방사선사는 바로 다름 행동을 취하지 않았을 것인데, 개발 당시에는 예측하지 못한 단순한 문제인데 실제 운영 상황에서는 심각한 문제로 이어지는 결과를 초래했다.

③ 제조기업의 무지

사고 발생 후 제조 기업에서는 소프트웨어에 대한 지나친 과신으로 검사를 제대로 하지 않았다. 당시 기술로는 시스템에 문제가 발생하였을 경우 하드웨어만 검사하여 원인을 찾기 힘들었고 코드에 대한 3자 리뷰가 없어 소프트웨어의 원인 탐색이 힘든 상황이었다. 결국, 소프트웨어의 신뢰성 및 안전성 개념의 무지로 인해 사고가 발생하였다.

3) 시사점

AECL은 Therac-25에 대한 수정 최종안을 제출했는데, 사고의 원인으로는 신기술에 기존 소프트웨어를 적용함에 있어 사전 평가가 부재한 점과 잘못 설계된 오류 및 경고 메시지, 그리고 자주 발생하지 않는 문제에 대한 기술자의 이해 및 수정이 불충분하다는 점을 보고했다. 승인을 위한 조건으로 턴테이블의 위치를 확인하는 센서 추가, 턴테이블 위치가 정확치 않을 때 X선 발사 금지, 불분명한 메시지를 의미가 있는 메시지로 변경, 사고방지를 막기 위한 작동자 버튼(Motion Enable FootSwitch) 추가 등 30가지 이상의 수정이 일어났으며, 이에 따른 매뉴얼을 변경하도록 하였다. 위험 분석을 통한 안전 메커니즘을 추가하고, 문제

점으로 밝혀진 매뉴얼 등을 수정한 것이다.

Nancy Leveson의 자료를 보면 일련의 사고 이 후 사고의 원인으로 다음과 같은 사항을 배우게 된 점이다.[27] 첫째 신뢰성과 안전성은 다르다. Therac-25는 문제가 생기기 전에 수만 번 사용할 정도로 신뢰가 있는 시스템이었고, ACEL은 이 시스템이 신뢰성이 있었기 때문에 안전하다고 생각했다.

둘째 안전을 확보하기 위해서는 방어적 설계가 필요하다. 소프트웨어의 오류를 검사하는 로직이 필요함은 물론 환자 자체의 상태를 살피는 것이 가장 안전을 확보하는 길이다. 설계자 및 개발자는 가장 나쁜 상황을 고려하여 시스템을 제작해야 한다.

셋째 위험을 제거할 때 전체 시스템을 분석하여 문제를 해결해야 한다는 것이다. Therac-25의 경우 방사선의 과다 노출 시, 그 부분의 문제만 해결하고 재가동을 했기 때문에 6명이나 인명피해를 입히게 되었다. 위험 분석은 전체 시스템에 대해 근본적인 해결책을 찾아야 한다.

네째 요인은 소프트웨어 개발자에 대한 것이다. Therac-25 개발자 소프트웨어 개발자는 다음 사항을 지키지 않았다.

- 소프트웨어 규격 및 매뉴얼은 즉시 작성하라
- 소프트웨어 품질 보장 및 표준을 지켜라
- 설계는 간단하게하고 위험한 코드는 피하라
- 오류 점검 및 검증은 개발 초기부터 하라
- 매뉴얼, 오류 정보 등 운영자에게 필요한 정보를 제공하라

다섯째는 소프트웨어 재사용에 관한 것이다. 재사용된 소프트웨어는 안전을 보장하지 않으며, 위험분석에 포함되어야 한다.

마지막으로 안전과 편리성에 대한 부분이다. 운영자가 편리하게하기 위해 데이터에 대한 정보 제공을 생략하고, 안전 점검이 자동으로 이루어지게 하면 안전에 위험이 될 수 있다.

소프트웨어 안전 확보를 위한 방안은 일반적인 안전 확보 방안과 소프트웨어에 특징지어진 확보방안이 있다. 일반적인 확보 방안으로는 안전에 대한 개념을 확실히 하고, 편리함 대신에 엄격한 안전 기준을 적용하는 것이다. 소프트웨어 관련된 안전 확보 방안은 전체적인 위험분석 후 설계, 개발, 검증 시 안전 확보 가이드 및 표준을 지키는 것이다.

---

27) N.G. Leveson, C.S. Turner(1993), “Medical devices The Therac25“, IEEE

2. 안전개발 설계 및 테스트 미흡으로 인한 사고 사례

자율주행차의 안전에 대해서는, 미 연방 자율주행차 정책(Federal Automated Vehicles Policy)의 美 자율주행차 성능 가이드라인과 차량전자제어시스템 안전표준 분석(전자 신뢰성 연구 프로그램(Electronics Reliability Research Program)’)에서 대해 조사된 바 있다.

현재, 자율주행차를 개발하기 위해 기존 완성차 업체는 물론, 구글과 같은 IT기반 기업들이 참여하고 있는데, 아직은 초기 모델 또는 서비스의 주행시험 단계라고 볼 수 있다. 각종 가이드라인과 표준을 적용함에도 다양한 사고들이 발생하고 있는데, 경미한 사고는 물론 서비스를 중단할 정도의 치명적인 사고도 발생하고 있어 업계의 우려를 낳고 있다. 다음은 우버 자율주행차 사고에 대한 사례이다.

1) 사고 경위

2018년 3월 19일에 미국의 차량공유업체 우버의 자율 주행 차량에 치여 보행자가 숨진 사고가 있었다. 당시 차에는 한 명의 운전자가 탑승하고, 컴퓨터 제어 모드에서 자가 운전 시스템으로 작동하고 있었다. 이 사고로 보행자는 치명적인 부상을 입고 사망했으나, 운전자는 부상을 당하지 않았다.

[그림 2-18] 사고가 난 위치 및 파손된 차량

(좌) Mill Avenue에서의 추돌 위치, (우) Uber 테스트 차량의 사고 후 파손상태

출처 : NTSB(2018), Preliminary Report Highway: HWY18MH010

이 사고 여파로 자율주행 기술을 연구하는 많은 기업들이 자체적으로 도로 주행 시험 등을 중단했다. 우버 또한 사고가 발생하자 애리조나 주 피닉스・템페와 피츠버그, 샌프란시스코, 토론토 등지에서 진행하던 자율주행차 시험 운행을 전면 중단했으나, 2018년 12월 자율주행차 시험주행을 재개하였다.[28)]

2) 사고 원인

이 사고의 원인으로는 ① 자율주행차에 적용된 소프트웨어로 동작하는 '불완전한 사물 인식 기능'과 함께, ② 비즈니스에서의 경쟁 우위를 점하기 위해 안전 기능을 배제한 채 실제 환경에서 테스트를 실시한 점 등을 들 수 있다.

① 불완전한 소프트웨어 기능

우버 자율 주행 사고 조사 결과 미국 고속도로교통국(NHTSA)은 자율주행 소프트웨어가 충돌 6초 전에 보행자를 발견했지만 단순한 물체 또는 다른 차로 인식하면서 사고가 발생했다고 발표했다[29)]. 이어 충돌 1.3초 전에 긴급 비상제동장치(EBS) 작동이 필요한 상황으로 판단했지만, 해당 기능이 우버에 의해 차단돼 결국 보행자를 충돌하는 결과가 나타났다. 우버는 자율주행 시 자동 비상제동장치가 잦은 오류를 일으켜서 비활성화를 시켰다.

당시 운전자가 보행자를 인식하고 핸들을 조작한 시간은 1초미만으로 나타났고, 사고를 방지할 수 없었다. 이는 불완전한 소프트웨어 기능과 운전자의 자율주행에 대한 과신이 복합되어 일어난 사고이다.

② 안전 기능(EBS)을 배제한 채, 실제 환경에서 테스트

앞에서 설명했듯이, 사물을 인식하는 소프트웨어가 불완전한 상태로는 안전한 주행을 확보할 수 없었을 것이다. 따라서 소프트웨어의 안전성을 높이기 위해서는 안전 방안으로서 EBS의 적용이 필수적이었으나, 우버 자율주행은 이를 배제한 채 운행했다.

소프트웨어 안전 프로세스에 따라 위험분석을 수행했다면, 불완전한 소프트웨어를 보완하기 위해서는 EBS의 적용이 필수적이었을 것이다. 그러나 이러한 분석결과를 무시한 채 소프트웨어를 운영했고, 제한된 구역이긴 하나 실제 환경에서 주행 테스트를 수행함으로써 치명적인 인명사고를 일으킨 것이다.

---

28) 뉴시스, http://www.newsis.com/view/?id=NISX20181221_0000509689&cID=10101&pID=10100
29) 오토타임즈, http://autotimes.hankyung.com/apps/news.sub_view?nkey=201805310802331

3) 시사점

자율주행차 사고는 최초의 사고인 테슬라 자율 주행 사고 이 후, 구글, 우버 등 자율주행자의 선도적인 업체들의 사고가 이어지고 있다.[30] 최근 자율주행차의 기술이 많이 향상되었으나, 이를 바라보는 사용자는 기대와 우려의 시각으로 바라보고 있다. 이는 기술의 문제도 있지만, 사용자의 안전에 대한 신뢰를 높이는 것도 중요하다. 테슬라의 경우는 사용자가 전방 주의 의무를 지키지 않아 일어난 사고로 책임을 운전자에게 미루고 있다. 우버의 경우도 자율주행시스템의 문제라기보다는 보행자의 잘못이라는 의견도 있다.

도요타 급발진 사고와 같이 자동차 사고는 운전자, 보행자, 다른 사고 차량 등 여러 요인에 의해 일어나기 때문에 그 요인을 밝히기가 어렵다. 이는 곧 위험분석이 다른 경우보다 복잡하고 어렵다고 해석할 수 있다. 그러나 자율주행차의 보급을 위해서는 안전이 최우선되어야 하며, 자율주행차는 이러한 위험 요인들을 모두 분석하여 안전을 확보해야 한다. 자율주행차의 운행 환경을 제한하여 안전을 확보하려는 시도가 있다. 예를 들면 자율주행차 전용 도로를 확보하는 것이다.[31]

미국, 영국 등 안전 선진국은 자율주행차 표준을 따로 규정하지 않고, 업체들에게 안전 확보를 보장하도록 규제를 풀어주고 있으며, 일본, 중국 등은 자율주행차 산업을 선점하려고 국가가 주도하여 안전 규정들을 제정하고 있다.[32] 이러한 법과 표준들을 기반으로 하는 안전 확보를 위한 중요한 시도가 될 것이다.

반면에 사고 시에 자율주행차에 안전 확보에 가장 주요한 요인의 하나인 소프트웨어에 원인을 찾기 어려운 것에 대해서는 이유를 찾아볼 필요가 있다. 이는 앞에서 언급했듯이 사고의 원인이 한 가지가 아니라 여러 가지이기 때문이기도 하고, 소프트웨어의 복잡성에 의한 이유도 있을 것이다. 아직까지는 자율주행차는 완전하지 않기 때문에 운전자에게 안전 확보의 의무를 같이 지고 있으며, 자율주행차의 상용화를 위해서는 이 문제가 해결되어야 할 것이다.

30) 진희승 et al.(2017), 자동차 산업의 SW안전 이슈와 해결과제
31) 진희승 et al.(2017), 자동차 산업의 SW안전 이슈와 해결과제
32) 진희승(2018) SW안전 산업동향조사

3. 61508 협회 및 CASS[33] Scheme 사례

1) 61508 협회 소개

(1) 개요

소프트웨어가 많은 분야에 적용되면서 소프트웨어를 포함한 안전 시스템이 더욱 복잡해지고 있다. 안전성 관련 시스템인 경우 안전성이 확보되었는지 증명을 요구하거나 확인할 수 있는 증거를 요구하고 있는 사례가 증가하고 있다.

IEC 61508은 참여자가 함께 작업하여 시스템이 요구된 수준의 안전 무결성을 유지하도록 보장하는 프레임워크를 제공한다. 때문에 안전한 제품을 만들고 운영하는데 참여하는 설계자, 개발자, 제조업체, 설치자, 운영자 또는 안전 시스템 관리자들은 IEC 61508을 잘 준수하고 활용할 수 있는 도움이 필요하게 되었다.

이에 61508 협회는 IEC 61508 및 관련 표준을 입증하고 실무에 적용하는데 도움을 주고자 만들어졌고, 협회 사이트(http://www.61508.org)를 방문하면 협회가 하는 일과 제공하는 자료를 누구든지 업무에 활용할 수 있도록 공유되어 있다.

협회는 다양한 실무그룹(WG)을 운영하고 있는데, WG은 필요에 따라서 특정 주제에 대한 실무에서 사용할 수 있는 IEC 61508 지침 및 가이드를 개발하고, 결과물은 협회 사이트에서 제공된다.

(2) 참여 멤버 및 역할

회원에는 최종 사용자, 시스템 통합자, 하위 시스템 및 부품 제조업체, 인증 컨설턴트 및 인증기관이 포함되며, 영국의 보건안전처(HSE, Health and Safety Executive)가 특별 회원으로 참여하고 있다. 회원들은 IEC 61508 표준에 맞는 업계 지침을 만드는데 우선적으로 참가하며, 규제 기관 및 평가 기관에 정책 및 의견을 제시한다. 협회 모임을 통해서 관련 전문가를 통해 지식 및 역량을 강화하고 멤버간 지식 공유 및 네트워킹으로 안전 표준 적용을 최적화 할 수 있다.

---

33) CASS : Conformity Assessment of Safety-related Systems

회원은 기능 안전을 달성하고 위험을 관리하기 위한 기준으로서 IEC 61508 및 관련 표준을 인정해야 하고, 안전 표준을 적용 할 때, 안전 수명주기를 준수해야 하며, 무결성, 투명성, 일관성 있는 방법을 사용하여 IEC 61508 및 관련 표준 준수 입증하는데 적극적으로 참여해야 한다.

[그림 2-19] 61508 협회 회원사

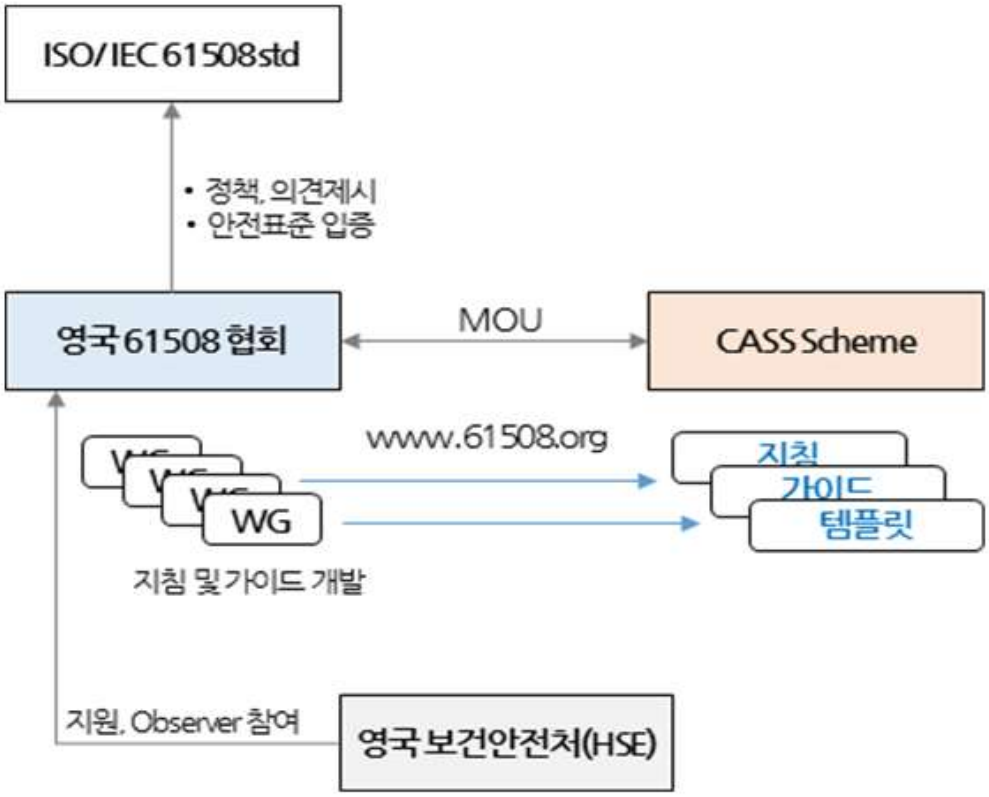
출처 : 출처 : https://www.61508.org/members/index.php 재정리

(3) 61508 협회와 CASS Scheme과의 관계

안전 시스템의 무결성을 입증하기 위해서는 조직의 기능적 안전 관리, 안전 시스템 자체 또는 시스템 구성 요소 등 모든 면에서 표준 준수를 증명해야 한다. 산업계에서는 영국정부의 지원을 받아 이를 위한 도구들을 제공하기 위해 CASS(안전 관련 시스템 적합성 평가, Conformity Assessment of Safety-related Systems) 제도를 시작하였다. 인증 적합성 평가 절차는 무료로 공개하고 투명하게 관리되며 평가자 및 피평가자들이 언제든 접근하여 사용할 수 있다. 이를 통해 인증 제공업체가 운영업체, 공급업체 및 규제기관이 신뢰할 수 있는 절차 및 기법을 통해 높은 수준의 안전 보증을 제공할 수 있게 되었다. CASS 제도의 관리에 대한 책임은 독립적인 이사회에 있으며, 61508 협회와 CASS 제도는 양해 각서[34] (Memorandum of Understanding)하에 함께 관리된다.

34) 출처 : https://www.61508.org/aboutus/index.php

[그림 2-20] 61508 협회 와 CASS Scheme Ltd.의 관계



출처 : https://www.61508.org 의 내용으로 자체 작성

새로운 평가 템플릿 또는 지침 문서는 양당사자가 서명한 양해각서(MOU)하에 61508 협회 내에 설립된 기술 워킹 그룹에 의해 작성된다. 따라서 61508 협회는 CASS의 주요 기술 자원을 제공한다.

2) CASS Scheme 소개

안전 분야의 선진국인 영국에서는 안전관련 시스템의 도입, 수정, 유지보수의 수명주기 단계별로 안전 활동을 수행하고 그 결과물을 국가가 지정한 기관에서 심사하도록 정부차원에서 강제하고 있다. 이러한 공인 기관의 심사를 위해서는 안전성 관련 요구사항 준수여부에 대한 객관적이고 효율적인 평가를 위해 공통의 양식이 필요하게 되었으며, 이러한 요구에 의한 안전성 관리 및 문서화 지침은 CASS(Conformity Assessment of Safety-related Systems)에 공유되어 많은 사업의 안전성 관리 문서화에 활용되고 있다.

CASS는 IEC 61508 및 관련 표준 준수 여부를 평가하기 위한 개방적 방법으로 그 핵심에는 IEC 61508 관련 조항에 준수증거(평가자의 문서)를 매핑하고 평가자가 해당 증거에 대한 평가를 기록하는데 사용할 수 있는 많은 평가 템플릿을 제공하는데 있다. 이 방법론은 다양한 평가를 통해 평가자에게 도움을 주고, 각 템플릿에는 특정 유형의 평가와 관련된 여러 평가 목표(TOE. Targets Of Evaluation)가 제공된다.

평가 템플릿은 무료이며 다운로드 페이지에서 다운받아 사용할 수 있으며, 템플릿과 함께 방법론을 구성하는 여러 산출물이 있어 외부 감사나 공인된 인증을 준비하는 피 평가자들이 자체적으로 대응을 위한 문서 양식으로도 사용할 수 있다.

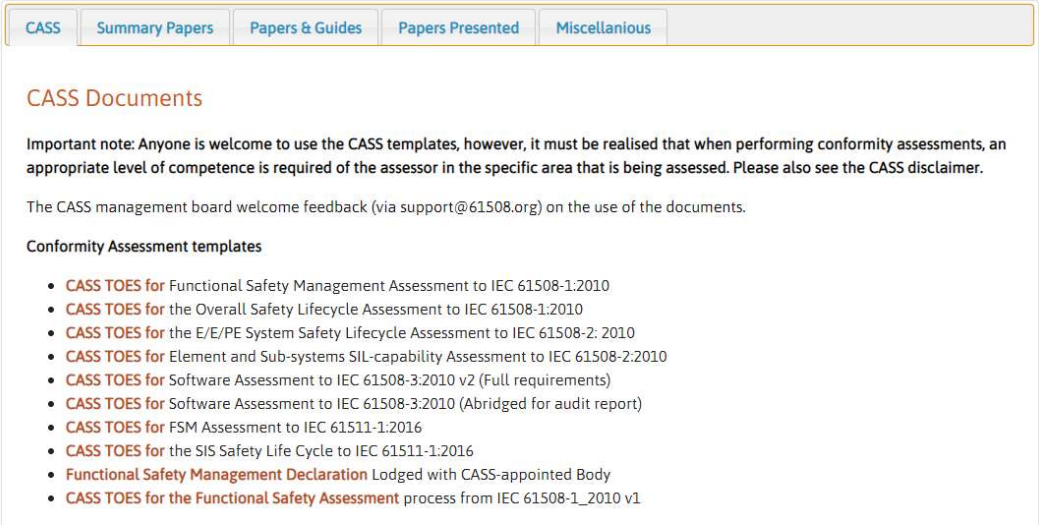
최종 사용자가 제시한 안전 요구사항의 만족을 입증하기 위해서는 수많은 입증자료가 요구되는데 이러한 입증 자료의 작성과정에서는 준비된 입증 자료의 최종 사용자 인정 여부가 실무에서 가장 어려움으로 나타나고 있다. CASS 방법론은 업계에서 기능 안전 준수를 입증할 수 있도록 설계되어 있다. 또한 모든 자료는 무료로 완전 개방되며, 투명한 평가 방법을 제공하고 공인된 방법론을 사용하여 피 평가자 조직에 적용할 경우 적합성을 외부에 제시 가능하며 국가 인증기관에서도 인정해 주고 IEC 61511과 같은 업계 지침 표준과의 호환성도 제시한다.

CASS Scheme Ltd의 회원은 영국 인증 기관 협회, 영국 컴퓨터 협회, 에너지 산업 협의회, 전자 산업 연합회, 영국 제어계측 자동화 무역협회(Gambica), 측정 제어 연구소, 화학 공학 연구소 (손실 방지 그룹), 엔지니어링 기술 대학, 철도 산업 협회, 61508 협회 등으로 구성되어 있다.

(1) CASS Documents 자료

누구든지 언제나 CASS 템플릿을 사용해도 되지만 적합성 평가를 수행할 때는 평가 대상 영역의 평가자는 적절한 수준의 역량을 가지고 있어야 한다. 이는 CASS 면책 조항에 제시되어 있다.

[그림 2-21] CASS Document 제공 내역



출처 : 61508 협회 홈페이지

(2) Summary Papers 자료

여기에서는 IEC 61508 및 관련 표준의 중요한 요구사항을 요약해서 알려준다. 내용은 특정 잠재고객 또는 특정주제를 대상으로 하며 IEC 61508에 정의된 Functional Safety에 대한 간략한 개요가 필요한 사람들을 대상으로 한다. 관리자, 운영자, 프로젝트 참여자, 구매자, 품질 담당자별로 문서를 제공한다.

(3) Papers & Guides 자료

논문 및 가이드는 기술적인 문제를 보다 심층적으로 고려할 필요가 있는 사람들을 대상으로 제공된다.

3) 시사점

표준들은 안전 시스템 확보를 위해 지켜야 하는 점을 포함하고 있으나, 그 항목들을 수행하는 자세한 방법은 포함하고 있지 않다. 61508 협회에서는 IEC 61508 표준을 실무에 적용하는데 도움을 주고자 만들어 졌으며, 관련기업들이 경험을 공유하고 있다. 국내에서도 이러한 활동을 위한 협회나 포럼이 필요하다.

4. 안전보건공단 사례

1) 공단 소개

안전보건공단(KOSHA, Korea Occupational Safety and Health Agency)은 근로자가 안전하고 건강하게 일할 수 있도록 하고 사업주가 재해예방에 힘쓰게 하여 국민경제발전에 기여하기 위하여 한국산업안전보건공단법에 의거 1987년 12월 설립된 고용노동부 산하 위탁집행형 준정부기관이다.

[그림 2-22] 산업안전보건공단 사업 안내

출처 : 산업안전보건공단 홈페이지

공단의 역할은 다음과 같다.

- 산업재해예방기술의 연구·개발 및 보급
- 산업안전보건에 관한 교육
- 사업장 산업재해예방을 위한 안전·보건 진단 또는 관리, 기술지원
- 유해하거나 위험한 기계·기구 등의 안전인증 또는 안전검사
- 산업재해예방을 위한 시설자금 지원
- 산업안전보건에 관한 정보 및 자료의 수집·발간·제공
- 산업안전보건에 관한 국제협력

2) 안전 관련 주요 제공 내용

(1) 안전 보건 자료실

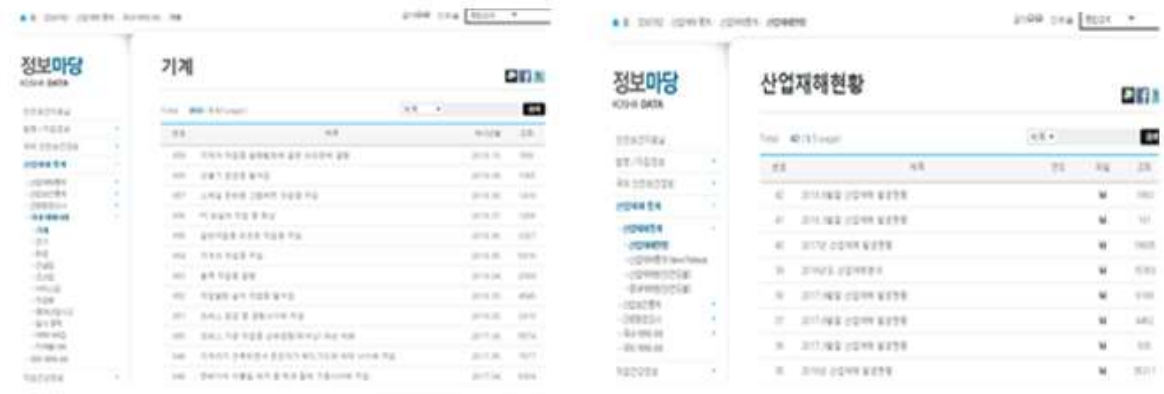
본 자료실에는 공단에서 발간되는 모든 미디어 자료를 한곳에 모아놓은 곳이다.

[그림 2-23] 안전보건자료실 제공 자료

출처 : 산업안전보건공단 홈페이지

산업재해현황, 산업재해통계, 산업재해 원인, 중대재해 원인에 대한 통계, 근로자 건강진단 결과, 근로환경 조사, 국내 재해사례, 국외 재해 사례 등의 통계를 제공한다.

[그림 2-24] 국내외 재해사례, 재해 통계



출처 : 산업안전보건공단 홈페이지

(2) 안전보건교육 포탈

경영층, 중간관리자, 노동자 등에 맞는 맞춤형 교육을 제공한다.

[그림 2-25] 안전보건교육포탈 제공 자료

출처 : 안전보건교육포털 홈페이지

(3) e-실무길잡이[35]

e-실무길잡이는 50인 미만 소규모 사업장 사업주 및 관리감독자를 기본 대상으로 해당업종에 맞는 안전보건 실무에 필요한 정보를 제공(업종특성, 공정·작업별 현황, 유해·위험요인, 법령정보, 현장 안전보건관리 사항, 재해사례, 체크리스트 등)하는 웹사이트. 근로자 중심의 안전보건 미디어 보급과 병행하여 산재예방·추진 의무가 있는 사업주에 대한 안전보건 정보 제공, 작업장에서의 안전보건 작업 공유를 통하여 산업안전보건법 준수 및 자율안전보건활동 활성화를 지원한다.

(4) 위험성 평가 지원시스템[36]

위험성평가 지원시스템(KRAS)는 사업장에서 의무적으로 실시하는 위험성 평가를 도와주는 시스템으로 위험성평가 실시, 인정심사 신청, 교육신청, 컨설팅 신정 등을 지원하며, 관련 컨설팅 기관 및 교육 기관을 안내한다. 위험성평가 우수사업장을 관리하고 있으며, 위험성평가 우수사업장 인정을 받을 시 산재보험료 20%인하, 인정유효기간(3년) 동안 정부의 안전·보건 감독 유예, 위험성평가 인정 시 클린 보조금 일천만원 추가 지원 등의 혜택을 주

35) http://guide.kosha.or.kr/guide/index.do
36) http://kras.kosha.or.kr/

고 있다.

위험성 평가에는 초보자에게 어려운 내용이 있으므로 동영상 강의를 통해 쉽게 접근 할 수 있는 방법을 제공해 주고 있다.

[그림 2-26] 위험성평가 동영상 강의 예시

출처 : 위험성평가 지원시스템

또한 업종별로 실제 작성 템플릿을 사례와 함께 제공해 주고 있다. 업종별로 템플릿과 예시가 잘 적혀있어 그 중에서 사업장에 해당되는 것을 선택하여 수정하면서 작성하면 되고, 유해 위험원 예시, 위험성 평가표, 감소대책 수립 및 실행 등의 항목으로 구성되어 있다.

3) 시사점

전통적인 안전 관리를 위해서는 법제도가 제정되고, 관리기관이 존재하여 위험성분석, 안전관리, 안전교육 등을 지원하고 있다. 고용노동부에서는 근로자의 안전관리, 행정안전부에서는 국가와 지방자체단체의 안전관리, 국토교통부의 항공, 철도 안전 관리 등 관련 부처에서 안전을 관리하고 있다.[37] 소프트웨어 안전에 관해서도 이러한 활동들이 수행되어야 한다. 기존 법과 안전 관리 체계를 참조하여, 소프트웨어 안전을 위하여 소프트웨어 안전 관리 체계를 구축하는 것이 필요하다.

37) 산업안전보건법, 재난 및 안전관리 기본법, 항공안전법, 철도안전법

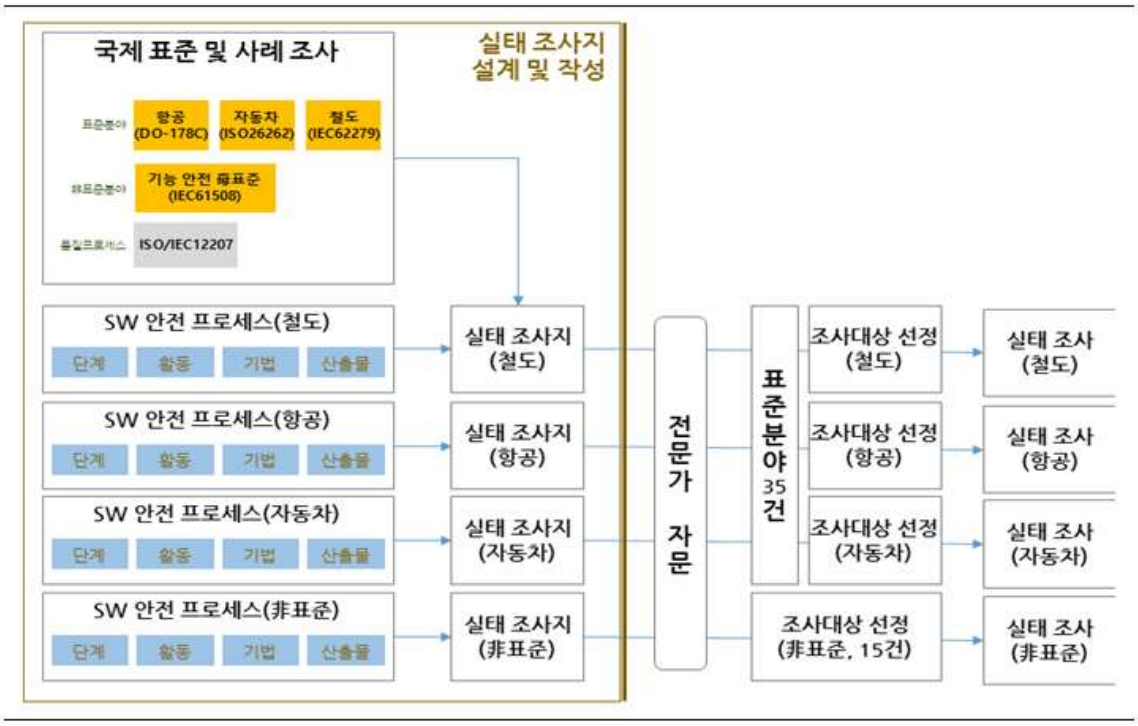
제3장 조사 설계 및 준비

제1절 개요

소프트웨어 안전 프로세스 실태조사는 먼저 실태 조사를 할 조사 항목에 대한 파악이 선행되어야 한다. 조사항목은 항공 분야는 DO-178C, 자동차 분야는 ISO/IEC 2626의 Part6, 철도 분야는 IEC 62279, 비표준 분야는 IEC 61508의 Part3을 기준으로 삼았다. 각 안전 표준에서는 먼저 상위 단계로 수명주기 단계를 정의하고, 향후 분석에 용이하도록 단계를 각 분야 간 서로 비교 가능할 수 있도록 정리하였다.

이후 단계별 세부 활동을 정의하였는데 이는 일반적인 활동과 안전 활동으로 나누었다. 일반 활동은 품질 관련 활동과 중복이 되는 개발 프로세스 활동이고 안전 활동은 위험분석, 안전 기능, 기법 등 안전과 관련된 활동으로 분류하였다. 이어서 분야별 사용하는 기법을 조사항목으로 선정하였는데, 이는 각 안전 표준별로 안전 무결성 등급별 적용해야 하는 기술 및 평가, 방법 등을 조사항목으로 선정하였고, 이어서 산출물에 대한 작성 및 관리 수준을 묻는 질문을 추가하였다.

[그림 3-1] 실태조사 준비 및 수행 절차



표준 및 비표준 분야별 실태 조사지를 작성한 후 분야별 전문가를 통한 자문을 실시하였다. 실태 조사지에 자문 의견을 반영하고 최종 실태 조사지를 생성하였다.

실태 조사를 위해서 조사 가능한 기업에 대한 대상기업을 확보하고 확보된 기업에 대한 사전 조사를 실시하였다. 소프트웨어 관련 기관의 기업 목록을 활용하여 표준 분야와 비표준 분야에 해당하는 기업 850여 건을 수집하였고, 수집된 기업에 대한 조사 가능 및 대상 적정성 여부를 조사하였다. 조사 대상 기업의 확보를 위해 소프트웨어정책연구소 보유한 기업, 정보통신산업진흥원 보유 한 기업 및 조선, 철도, 자동차, 항공 관련 협회 회원사 목록 등을 중심으로 수집하였다. 소프트웨어 안전성 조사의 가능성 및 적정성을 판단하기 위해 대상 기업에 대한 기능 안전성 관련 여부, 소프트웨어 개발 여부, 안전 관련 조직 보유 여부, 안전 담당자 보유 여부, 조사 대상 분야 연관성 여부 등을 기준으로 선정하여 최종 119개 기관을 조사 대상으로 선정하였다.

1차 선정된 기관에 대해 전화 접촉을 실시하여 실태 조사 및 심층 조사를 수행하였다. 조사 수행을 위해 조사 기관에 사전 전화 연락을 통해 실태 조사 일정을 확정하고 조사원에 대한 사전 교육을 실시 한 후 방문 조사를 통해 조사지 및 심층 분석을 실시하여 비표준 15건, 항공분야 13건, 철도 11건, 자동차 11건의 총 50건의 조사를 수행하였다.

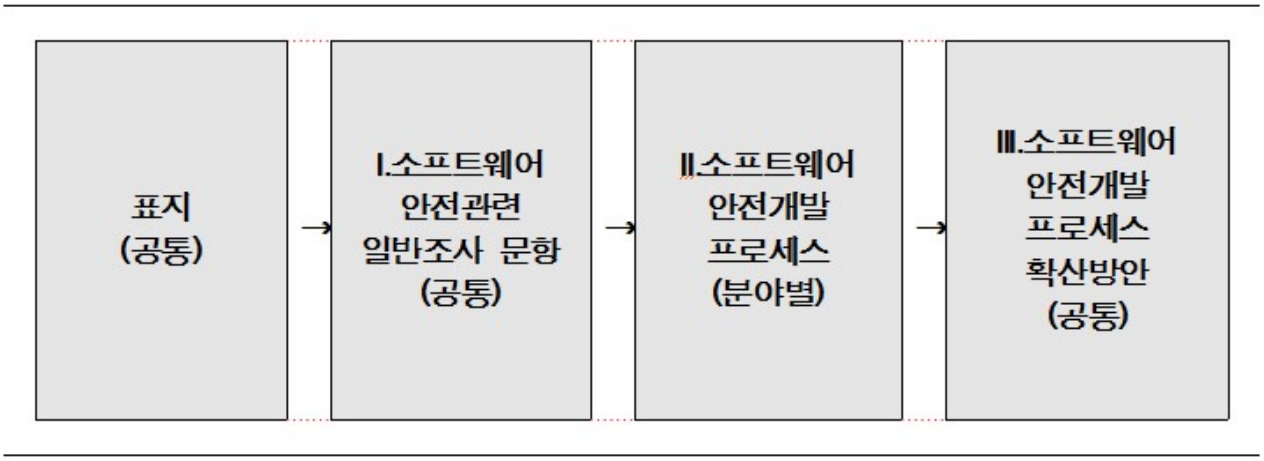
〈표 3-1〉 실태조사 조사 관리 현황

도메인		대상	전화 접촉			접촉 결과		조사 확정	조사 완료
			접촉	잔여	접촉율	거부	연락불가	건수	건수
비표준	조선기타	28	28	0	100%	7	6	15	15
	계							15	15
표준	항공	24	18	6	75%	2	1	15	13
	철도	17	17	0	100%	4	1	12	11
	자동차	40	26	14	65%	12	2	12	11
	계							39	35

제2절 조사 항목 설계

실태조사지는 표준 분야와 비표준 분야로 구분하여 조사지를 정의하였다. 조사지 구성은 조사 대상 기업 현황, 표준 및 비표준 분야에 대한 공통 설문, 각 산업별 소프트웨어 안전 개발 프로세스, 확산방안 도출을 위한 향후 개선 및 지원 요구사항 등을 조사지로 구성하였다.

[그림 3-2] 실태 조사지 구성



조사지의 항목 설계는 첫 번째, 표지는 기업명, 주소, 종업원 수, 기업 규모, 주력 산업 분야, 매출액, 주요제품, 산업 인증, 소프트웨어 비중, 작성자 현황을 조사지 항목으로 설계하였다. 두 번째, 공통부분은 표준 및 비표준 분야에 대한 산업 전체에 대한 안전 일반 현황, SW안전 제품 현황, 개발 시 지원활동 등 각 산업 공통항목을 포함하였다. 세 번째는 각 산업별로 표준에 기반으로 하여 요구사항 정의, 설계, 개발, 검증의 단계에서 수행하는 활동들에 대해 조사하였다. 각각의 단계는 동일하나, 표준별로 사용하는 활동, 기법 등의 세밀한 부분은 설문지에 포함하여 조사의 객관성을 유지하고, 사용자들에게 대답의 지침을 제공하였다. 세세한 설문 대용은 설문 결과에서 소개한다. 마지막으로 소프트웨어 안전 개발 프로세스를 위한 확산을 위한 문제점, 인식, 정책지원 사항에 대해 조사하였다.

표준 분야에는 자동차, 항공, 철도 분야의 국제 표준을 기반으로 소프트웨어 안전 개발 프로세스에 대한 조사 항목을 구성하였고, 비표준 분야는 IEC 61508을 기준으로 소프트웨어 안전 개발 프로세스에 대한 조사 항목을 구성하고 설계한다. 본문에서는 조사 설계 내용을 간략히 제시하고 설문지는 부록에 제시한다.

1. 조사지 공통 구성 항목

각 산업별 조사에 앞서 공통 구성 항목을 SW안전 일반 현황, 제품 현황, 개발 프로세스 개요, 위험분석 방법, 지원 항목 등 공통으로 질의가 가능한 항목들로 구성하였다. SW안전 일반 현황에서는 표준과 발주사 개발 요건 및, 개발사 프로세스 적용 경험에 대해 조사하였다. 제품 현황에서는 조사 기업의 SW안전 관련 제품과 안전성 확보 이유를 조사 항목에 포함시켰다.

개발 프로세스 개요에서는 안전 담당자의 SW안전 프로세스 참여 형태, 프로세스 적용 미비 이유, 안전계획, 산출물 관리 등을 조사하였다. 응답자의 편의 및 조사의 객관성을 위하여 질문지에 대답이 가능한 사례를 추가하였으며, 추가로 응답자의 자유로운 의견을 표출하도록 유도하였다.

SW를 포함한 주요제품 개발을 위한 사내외 위험분석 방법 활용 여부와 위험 분석 기법 등을 조사하고, 관련 제품 등도 조사하였다. 위험 분석 후 안전 무결성 등급(SIL, Safety Integrity Level) 결정 방법도 조사항목에 포함하였다.

SW 개발 시 지원활동으로는 형상관리와 변경 요청 및 관리에 대해 항목에 추가하였다. 현상관리 및 요구사항 관리 도구에 대해서도 조사하였다.

〈표 3-2〉 공통 조사 항목

구분	항목	실태 조사항목(설문 문항)
SW안전 일반 현황	표준	국내외 관련 표준 및 발주사 개발 요건에 명시 여부
	경험	안전 프로세스 적용 경험 및 이해 수준
제품현황	제품	안전성 확보가 필요한 제품 및 이유
개발 프로세스	참여형태	안전 전문가 프로세스 참여형태
	안전계획	안전계획 수립 여부 및 미 수립 시 이유
	산출물	위험 분석 수행 산출물 관리 여부 및 미비 이유
위험 분석	기법	위험 분석 방법 및 기법, 도구
	SIL	안전무결성 등급(SIL) 결정 방법 및 제품
지원활동	형상관리	형상 관리 및 미비 이유
	변경관리	변경 관리 및 도구

2. 자동차 분야 안전개발 프로세스 조사지 설계

1) 조사 설계 세부 항목 개요

자동차 분야의 안전 국제 표준인 ISO 26262 국제 표준을 참조하여 조사지 항목을 설계하였다. 조사지 설계를 위해 소프트웨어 안전 개발에 대한 프로세스를 정의하였다. 자동차 분야의 소프트웨어 안전 개발 프로세스는 소프트웨어 안전 요구사항 관리, 소프트웨어 아키텍처 설계, 소프트웨어 유닛(단위) 설계, 소프트웨어 유닛 구현, 소프트웨어 통합 검증으로 프로세스를 확정하였다.

2) 조사 설계 세부 항목 내용

자동차 분야의 소프트웨어 안전 개발 프로세스 조사 항목은 5단계로 구성한다. 첫 번째 단계로는 소프트웨어 안전 요구사항 명세로서 활동, 기법, 산출물의 유형으로 조사지를 설계한다. 활동으로는 요구사항 명세 시 개발 및 품질 활동과 안전 요구사항 명세 시 사전 계획 수립을 어떻게 수행하고 있는지를 조사하기 위한 활동에 관한 내용을 구성하였다. 소프트웨어 안전 요구사항 명세를 수행하기 위한 기법에 대한 조사를 위해 안전성 관점의 요구분석 명세 기법은 무엇이 있고 어떻게 활용하는지에 대한 내용을 조사지 항목으로 구성하였다. 또한, 소프트웨어 안전 요구사항 산출물 조사를 산출물 작성 유무와 구성 내용에 대한 조사지 항목으로 구성하였다.

두 번째는 소프트웨어 아키텍처 설계에 대한 조사를 위해 활동, 기법, 산출물을 조사 유형으로 구성하였다. 아키텍처 설계 시 수행 활동과 소프트웨어 아키텍처 설계 단계의 활동 조사를 위해 소프트웨어 아키텍처 안전 분석 수준, 소프트웨어 아키텍처 설계 및 방법에 대한 조사를 위한 조사지 항목 구성하였다. 소프트웨어 아키텍처 기법 조사를 위해 아키텍처 설계 시 사용하는 표기법과 아키텍처 설계 검증 방법에 대해 조사 항목을 구성하였다. 소프트웨어 아키텍처 설계 단계에 어떤 산출물 작성 여부와 작성되는 내용이 무엇인지 조사하기 위한 내용으로 조사 항목을 구성하였다.

세 번째는 소프트웨어 유닛(단위) 설계에 대한 조사를 위해 방법으로 활동, 기법, 산출물에 대한 유형을 구분하여 조사지를 구성하였다. 소프트웨어 유닛 설계 단계의 활동을 조사하기 위해 유닛 설계 시 수행 활동에 대한 조사, 유닛 설계 시 사용하는 표기법 조사, 소프트웨어 유닛 설계의 설계에서 가독성, 단순성, 강건성 등 속성 설계 현황을 확인하기 위한 내용에 대한 조사를 조사지 항목으로 구성하였다. 기법 조사를 위해 설계 검증 기법은 어떻

게 적용되고 어떤 내용인지에 대한 조사를 위해 조사지를 구성하였다. 유닛 설계 수행 관련 산출물 작성의 여부와 어떤 내용으로 구성되어있는지 조사하기 위한 내용으로 조사지 항목으로 구성하였다.

네 번째는 소프트웨어 유닛 구현 측면에서는 구현 시 수행 활동이 어떻게 진행되고 있는지 확인하기 위해 조사지 항목을 정의하였고, 활동에 대한 조사를 위해 소프트웨어 유닛 구현에 대한 적합한 시험 환경 구성을 조사하기 위해 조사지 항목을 정의하였다. 안전 기법을 조사하기 위해 소프트웨어 유닛 시험 방법에 대한 조사지 구성과 소프트웨어 유닛 시험을 위한 시험 케이스 도출 방법을 확인하기 위한 내용을 조사지 항목으로 구성하였다.

마지막으로 소프트웨어 통합 검증 조사를 위해 활동, 기법, 산출물의 유형으로 조사지를 구성하였다. 소프트웨어 통합 검증에 대한 활동 조사를 위해 소프트웨어 통합 및 시험을 어떻게 진행하고 있는지 확인하기 위한 내용을 조사지로 구성하였고, 소프트웨어 안전 기능을 확보를 위해 통합 및 시험을 어떻게 진행하고 있는지 확인하기 위한 내용을 조사지 항목으로 구성하였다. 또한 산출물에 대한 내용과 형식을 조사하기 위해 시험 단계 수행 관련 산출물 작성 여부 및 산출물 내용을 조사하기 위한 내용으로 조사지 항목을 구성하였다.

<표 3-3> 자동차 분야 안전개발 프로세스 조사항목

프로세스	유형	실태 조사항목(설문 문항)
SW안전 요구사항 명세	활동	안전 요구사항 명세 시 안전 관련 활동
	기법	안전성관점 요구분석 명세기법
	산출물	안전 요구사항 명세 관련 작성하는 산출물 목록
SW 아키텍처 설계	활동	소프트웨어 아키텍처 수준 안전 분석, SW 아키텍처 설계 방법
	기법	아키텍처 설계 검증 방법, 아키텍처 설계 시 사용하는 표기법
	산출물	산출물 목록
SW유닛(단위)설계	활동	유닛 설계 시 일반적인 수행 활동, 언어
		소프트웨어 유닛설계의 설계 속성을 달성하기 설계 원칙준수
	기법	SW 유닛(단위) 설계 검증 기법
	산출물	단위 설계 수행 관련 산출물
SW유닛 구현	활동	구현 시 일반적인 수행 활동
		시험 케이스 완전성 평가를 위한 요구사항 구문 커버리즈 측정 방법
		소프트웨어 유닛 구현에 대한 적합한 시험 환경
	기법	SW유닛(단위) 시험방법
	산출물	SW 유닛 구현
SW통합 검증	활동	SW 안전기능을 확보를 위한 통합 및 시험
		시험 케이스 완전성 평가를 위한 요구사항 구문 커버리즈 측정 방법
	기법	소프트웨어 안전 요구사항 검증 실시를 위한 환경 구현
		HW, SW 통합 검토시 시 요구되는 기법
	산출물	시험 단계 수행관련 산출물 보유 및 사용 여부

3. 철도 분야 안전개발 프로세스 조사 설계

1) 조사 설계 세부 항목 개요

철도 분야의 안전 국제 표준인 IEC 62279 표준을 참조하여 조사지 항목을 설계하였다. 조사지 설계를 위해 철도 분야에 대한 소프트웨어 안전 개발에 대한 프로세스를 정의하였다. 철도 분야의 소프트웨어 안전 개발 프로세스는 요구사항 정의, 아키텍처 설계, 컴포넌트 설계, 컴포넌트 구현, 테스트의 프로세스를 정의하고 각 프로세스 단계별 상세하게 조사지 항목을 구성하였다.

2) 조사 설계 세부 항목 내용

요구사항 정의 단계에서 수행하는 활동은 요구사항 정의, 요구사항 적합성 검토, 종합 소프트웨어 테스트 명세 작성, 요구사항 검증이 있다. 이들 중, 요구사항 정의 및 요구사항 검증 활동에 대해서 안전성 관점에서 수행하는 활동을 조사하고, 그 외의 활동에 대해서는 일반 활동에 대해 조사한다. 위 활동이외에 요구사항 정의에서 적용하는 기법과 종합 소프트웨어 테스트 명세에서 적용하는 기법에 대해 조사하고, 마지막으로 요구사항 정의 단계에 작성하는 산출물에 대한 조사항목을 정의하였다.

아키텍처 설계 단계에서 수행하는 활동은 아키텍처 명세, 인터페이스 명세, 소프트웨어 설계, 소프트웨어 통합 테스트 명세, 소프트웨어/하드웨어 통합 테스트 명세, 아키텍처 및 설계 검증이 있다. 이들 중, 아키텍처 명세 활동에 대해서 안전성 관점에서 수행하는 활동을 조사한다. 위 활동이외에 아키텍처 명세 및 설계 시 적용하는 기법과 소프트웨어 통합 테스트 명세와 소프트웨어/하드웨어 통합 테스트 명세에 적용하는 기법에 대해 조사하고, 마지막으로 아키텍처 설계 단계에 작성하는 산출물에 대한 조사항목을 정의하였다.

컴포넌트 설계 단계에서 수행하는 활동은 컴포넌트 설계 명세, 컴포넌트 설계 적합성 검토, 컴포넌트 테스트 명세, 컴포넌트 설계 검증이 있다. 컴포넌트 설계 단계의 활동에 대해서는 일반적인 관점에서 수행하는 활동을 조사한다. 위 활동이외에 컴포넌트 설계 명세 시 적용하는 기법에 대해 조사하고, 마지막으로 컴포넌트 설계 단계에 작성하는 산출물에 대한 조사항목을 정의하였다.

컴포넌트 구현 단계에서 수행하는 활동은 컴포넌트 구현, 컴포넌트 테스트, 컴포넌트 테스트 결과 평가가 있다. 컴포넌트 구현 단계의 활동에 대해서는 일반적인 관점에서 수행하는

활동을 조사한다. 위 활동이외에 컴포넌트 구현 시 적용하는 기법에 대해 조사하고, 마지막으로 컴포넌트 구현 단계에 작성하는 산출물에 대한 조사항목을 정의하였다.

테스트 단계에서 수행하는 활동은 소프트웨어 통합, 소프트웨어와 하드웨어 통합, 통합 검증, 종합 소프트웨어테스트, 소프트웨어 확인이 있다. 테스트 단계의 활동에 대해서는 일반적인 관점에서 수행하는 활동을 조사한다. 위 활동이외에 소프트웨어 통합 시 적용하는 기법과 소프트웨어/하드웨어 통합 시 적용하는 기법에 대해 조사하고, 마지막으로 테스트 단계에 작성하는 산출물에 대한 조사항목을 정의하였다.

<표 3-4> 철도 분야 안전개발 프로세스 조사항목

단계	유형	실태 조사항목(설문 문항)
요구사항정의	활동	안전성 관점에서 요구사항 정의 및 검증 시 수행 활동
		요구사항 적합성 검토 및 종합 SW 테스트 명세 시 수행 활동
	기법	요구정의 및 종합 SW 테스트 명세 시 적용하는 기법 & 대책
	산출물	요구정의단계 산출물 목록
아키텍처설계	활동	아키텍처, 인터페이스 명세 시 일반적인 수행 활동
		아키텍처 명세 시 안전관련 수행 활동 (특히 SIL 3,4의 경우)
		SW 통합, SW/HW 통합 테스트 명세시 수행 활동
		아키텍처 및 설계 검증 시 수행 활동
	기법	아키텍처 명세시 적용하는 기법 & 대책
		설계시 적용하는 기법 & 대책
		SW, SW/HW 통합 테스트 명세 시 적용하는 기법 & 대책
	산출물	아키텍처 설계단계 산출물 목록
컴포넌트설계	활동	컴포넌트 설계 명세 및 검증 시 수행 활동
		컴포넌트 테스트 명세 시 수행 활동
	기법	컴포넌트 설계명세 시 적용하는 기법 & 대책
	산출물	컴포넌트 설계단계 산출물 목록
컴포넌트구현	활동	컴포넌트 구현 및 테스트, 평가 시 수행 활동
	기법	컴포넌트 구현시 적용하는 기법 & 대책
	산출물	컴포넌트 구현단계 산출물 목록
테스트	활동	SW, SW/HW 통합 및 검증, 종합 SW 테스트 시 수행 활동
	기법	SW, SW/HW 통합 시 적용하는 기법 & 대책
	산출물	테스트 단계 산출물 목록

4. 항공 분야 안전개발 프로세스 조사 설계

1) 조사 설계 세부 항목 개요

항공 분야의 소프트웨어 안전 표준인 DO-178C를 참조하여 조사지 항목을 설계하였다. 조사지 설계를 위해 항공 분야에 대한 소프트웨어 안전 개발에 대한 프로세스를 정의한다. 항공 분야는 소프트웨어 안전 개발 프로세스는 요구사항 정의, 설계, 코딩, 통합, 테스트의 프로세스를 정의하고 각 프로세스 단계별 상세하게 조사지 항목을 구성하였다. DO-178의 경우는 다른 표준과는 달리 아키텍처 설계와 상세설계와 구분되어 있지는 않으나, 다른 설문지와 결과를 비교 분석하기 위해 동일하게 설문지를 구성하였다.

2) 조사 설계 세부 항목 내용

소프트웨어 안전개발 프로세스와 관련하여 요구정의, 설계, 구현, 테스트로 나누어 조사항목을 구성하였으며, 조사 항목의 내용은 다음과 같다.

요구사항 정의 단계에서 수행하는 활동은 요구사항 정의, 요구사항 검증이 있다. 요구사항 정의 및 요구사항 검증 활동과 요구사항 정의 시 작성하는 산출물에 대한 조사 항목을 정의하였다.

설계 단계에서 수행하는 활동은 아키텍처 설계, 상세 요구사항 설계에 대한 조사지를 구성하였다. 아키텍처 설계에 대한 활동은 아키텍처 검증 시 수행하는 활동을 포함하였으며, 상세 요구사항 설계의 상세 요구사항 검증 시 수행하는 활동을 포함하여 조사 항목을 정의하였다. 설계 작성 시 필요한 산출물에 관한 내용을 조사 항목으로 정의하였다. 조사지는 철도, 자동차 등의 조사지와 동일하게 아키텍처 설계와 상세 설계를 구분하여 구성하였다.

구현 단계에서 코딩을 위한 활동을 조사하고 소스코드 검증 활동에 관한 내용을 조사 항목을 정의하였다. 소스코드 데이터와 제어 흐름 표현 여부, 표준 준수여부, 요구사항을 만족하는 지 여부 등을 예제로 제공하여 사용자들이 응대하는데 편리하게 구성하였다.

테스트 단계에서 수행하는 활동은 활동과 기법으로 구분하여 조사지를 구성하였고, 주요 내용으로는 테스트 설계, 테스트 수행, 테스트 검증에 관한 내용을 조사지 항목으로 구성하였다. 항공 분야는 비교적 표준을 잘 적용하는 분야기 때문에 구체적인 테스트 방법과 테스에 기법을 시행여부도 설문지에 포함하였다.

<표 3-5> 항공 분야 안전개발 프로세스 조사항목

구분	유형	설문문항
요구정의	활동	요구사항 정의 시 안전 요구사항으로 정의하는 항목
		요구사항 검증 시 수행 활동
	기법	요구사항 명세 시 사용하는 기법
	산출물	요구사항 정의 시 일반적으로 정의하는 항목
설계	활동	아키텍처 설계 및 검증 시 일반적인 수행 활동
		아키텍쳐 설계 시 안전을 고려하여 정의하는 항목
	기법	아키텍처 설계 시 안전을 위해 사용하는 기법
	산출물	설계시 정의하는 항목
상세설계	활동	상세 요구사항 설계 및 검증 시 일반적인 수행 활동
		상세 요구사항 설계시 안전 요구사항 반영 활동
	기법	상세 설계 검증 기법
	산출물	상세 설계 산출물 포함 항목
구현	활동	소스코드 구현 시 수행 활동
	기법	구현 단계에서 안전을 위해 적용하는 기법
	산출물	구현 단계에서 작성하는 산출물
테스트	활동	테스트 설계 시 수행활동
		테스트 환경 구성 시 수행활동
	기법	안전을 위해 사용하는 테스트 기법
	산출물	테스트 단계의 산출물

5. 비표준 분야 안전개발 프로세스 조사 설계

1) 조사 설계 세부 항목 개요

비표준 분야인 경우 기능 안전 모표준인 IEC 61508을 기준으로 조사지 항목을 설계하였다. 조사지 설계를 위해 소프트웨어 안전 개발에 대한 프로세스를 정의하였다. 비표준 분야이기 때문에 모표준인 IEC 61508과 일반적인 소프트웨어 표준 개발 프로세스를 참조하여 조사지 항목을 구성하였다. 비표준 분야의 소프트웨어 안전 개발 프로세스는 소프트웨어 요구사항 명세, 소프트웨어 아키텍처 설계, 소프트웨어 상세 설계, 소프트웨어 구현, 소프트웨어 시험으로 구성하고, 각 프로세스 단계별 상세하게 조사지 항목을 구성하였다. 비표준 분야의 설문지는 가능한 전문 용어를 제외하고 가능한 쉬운 용어로 변경하여, IEC 61508을 모르는 응답자도 설문에 대답하게 쉽게 구성하였다.

2) 조사 설계 세부 항목 내용

소프트웨어 요구사항 정의 단계는 IEC 61508 단계에서 소프트웨어 안전 요구사항 명세와 소프트웨어 안전 확증 계획 부분으로 구성하였다. 이때 활동으로는 소프트웨어 요구사항 명세 시 수행하는 안전관련 활동과 소프트웨어 안전 확정 계획 수립 시 수행하는 활동에 대한 조사를 수행하고 기법에서는 안전선 관점에서 요구분석 기법 활용 여부를 묻는데 관련 기법은 IEC 61508 Part3 ‘Table A.1 - Software safety requirements specification’ 의 기법을 알고 있는지를 조사하고자 한다. 이어서 산출물은 안전 요구사항 명세 관련 산출물로 IEC 61508 Part1 ‘Table A.3 - 소프트웨어 안전수명주기에 관련된 정보의 문서구조’ 의 요구사항 관련 산출물의 작성 수준을 조사하도록 구성하였다.

소프트웨어 아키텍처 설계 단계의 활동은 요구되는 안전 무결성 수준에 관련하여 소프트웨어 안전을 위해 명세된 요구사항을 충족하는 소프트웨어 아키텍처를 개발, 소프트웨어에 할당된 요구사항을 검토 및 평가, 개발 및 검증을 위해 사용하는 언어 및 도구에 대한 조사를 수행하고 기법에서는 Part3 ‘TableA.2 - Software architecture design’ 에서 제시하는 기법에 대한 인식 여부를 조사한다. 이어서 산출물은 Part1 ‘Table A.3 - 소프트웨어 안전수명주기에 관련된 정보의 문서구조’ 의 아키텍처 관련 산출물의 작성 수준을 조사하도록 구성하였다.

소프트웨어 상세 설계 단계에서는 요구되는 안전 무결성에 따라 소프트웨어 안전을 위해 명세된 요구사항을 만족하는 세부 설계 활동을 수행하는지를 조사하고 기법으로는 Part3

‘Table A.4 - Software detailed design’, ‘Table B.7 - Semi-formal methods’ 등의 알고 있거나 사용하는지를 알아보고자한다. 이어서 산출물은 Part1 ‘Table A.3 - 소프트웨어 안전수명주기에 관련된 정보의 문서구조’ 의 상세 설계 관련 산출물의 작성 수준을 조사하도록 구성하였다.

구현 단계에서는 안전 무결성에 따라 소프트웨어 안전 요구사항에 충족하는 소프트웨어를 설계하고 개발하는 활동을 조사하고, 기법은 Part3 ‘Table B.1 - Design and coding standards’ 의 기법의 알고 있거나 사용하는지를 알아보고자 한다. 이어서 산출물은 Part1 ‘Table A.3 - 소프트웨어 안전수명주기에 관련된 정보의 문서구조’ 의 구현 관련 산출물의 작성 수준을 조사하도록 구성하였다.

소프트웨어시험 단계에서는 소프트웨어 모듈시험, 소프트웨어 통합시험, 하드웨어와 소프트웨어 통합시험 시 수행하는 활동에 대한 조사활동의 수행 여부를 조사하고, 기법은 Part3 ‘Table A.5 - Software module testing and integration’, ‘Table A.6 - Programmable electronics integration (hardware and software)’ 등의 기법을 알고 있거나 사용하는지를 알아보고자 한다. 이어서 산출물은 Part1 ‘Table A.3 - 소프트웨어 안전수명주기에 관련된 정보의 문서구조’ 의 시험 관련 산출물의 작성 수준을 조사하도록 구성하였다.

〈표 3-6〉 비표준 분야 안전개발 프로세스 조사항목

프로세스	유형	실태 조사항목(설문 문항)
SW 요구사항 명세	활동	요구사항 명세 시 안전관련 수행 활동
		SW 안전 확증 계획 수립 여부
	기법	안전성 관점 요구분석 기법 활용 여부 및 예시
	산출물	안전 요구사항 명세 관련 작성하는 산출물 목록
SW 아키텍처 설계	활동	아키텍처 설계 시 일반적인 수행활동
		아키텍처 설계 시 안전관련 수행활동
		안전 무결성을 유지할 명확한 아키텍처 표현법 보유 여부
	기법	아키텍처 설계 시 사용하는 기법
	산출물	아키텍서 설계서에 기록되는 내용
SW상세	활동	상세 설계 시 일반적인수행활동

설계		SW 안전기능을 확보를 위한 상세설계 요구사항
		안전 무결성 요구사항 충족을 보증할 단위/통합 시험 계획 수립
		개발 사용 도구의 안전성 검증 여부(Tool Qualification)
	기법	안전 무결성 수준에 따른 설계 기법 사용 여부
		정형화된 모델링 기법의 사용
	산출물	상세 설계 수행관련 산출물 보유 및 사용 여부
SW구현	활동	구현시 일반적인 수행활동
		각 SW 모듈의 소스코드에서 검토해야 할 것들
SW시험	활동	시험시 일반적인 수행활동
		SW 안전기능을 확보를 위한 시험 단계 요구사항
	기법	모듈/통합 시험 기법
		HW, SW 통합 검토시 시 요구되는 기법
	산출물	통합 시험 명세에 포함되는 것
		시험 단계 수행관련 산출물 보유 및 사용 여부

제4장 실태조사 및 분석

제1절 조사 개요

산업별로 소프트웨어 의존성과 복잡성이 증가함에 따라 소프트웨어의 결함으로 인한 사고의 발생 가능성이 높아지고 있어. 국민 안전 확보를 위한 핵심으로 소프트웨어 안전성이 중요한 화두로 부상하고 있다. 본 실태 조사는 안전과 관련되는 제품에 소프트웨어가 포함될 경우 소프트웨어 안전 프로세스 적용이 소프트웨어 안전 확보에 필수적이기 때문에 산업별 소프트웨어 안전 프로세스 적용 실태 조사를 위한 목적으로 현황 조사를 실시하였다. 산업별 표준과 비표준으로 구분하여 조사를 수행하였고, 산업별 소프트웨어 안전 개발 프로세스 활동이 안전 수준과 안전에 미치는 영향을 분석하였다.

소프트웨어 안전 프로세스 실태 조사를 위해 항목, 수준, 대상 선정을 사전에 정의하고 산업 표준이 있는 항공, 자동차, 철도인 경우 각 표준에 기초로 하여 프로세스 적용에 대한 상세 항목을 설계하였다. 또한 소프트웨어 안전 개발 프로세스 미적용에 대해서는 적용이 어려운 이유를 조사하였다. 소프트웨어 안전 개발 프로세스 확산을 위한 현황 및 요구사항을 파악하기 위한 항목을 추가하여 조사 분석을 수행하였다. 비표준 산업 분야는 소프트웨어 안전 개념 및 적용 분야에 소프트웨어 안전 중요도에 대한 정도와 기존 소프트웨어 안전 프로세스 적용 시 문제점이 있는지에 대한 현황을 파악하고 실태를 조사하였다.

심층 조사 분석을 위해 소프트웨어 안전성에 대한 업무를 수행하고 있는 기업을 대상으로 진행하였으며, 소프트웨어 안전에 대한 소프트웨어 사업 현황, 인력 구성, 제품 역량, 연구소 역량, 개발 기술 역량 등을 기본적으로 파악한 후 산업별 심층 분석 업체를 선정하여 진행하였다. 소프트웨어 안전 프로세스 심층 실태 조사를 위해 조사지 설계는 비표준인 경우 IEC 61508 표준에 의거하여 해당 분야 전문가의 의견과 토론을 통해 심층 조사지를 설계하였고, 항공 분야는 DO-178C 표준, 자동차 분야는 ISO 26262, 철도는 IEC 62279 기반으로 심층 조사지를 설계하였다.

제2절 실태 조사 결과 및 분석

1. 설문 조사 응답자 특성

심층 실태조사를 위해 산업별 비표준 분야 산업, 자동차, 철도, 항공 등 소프트웨어 안전 프로세스를 적용하고 있는 기업 50개를 선정하여 심층 실태 조사를 실시하였다. 실태조사에 참여한 표준 산업 분야와 비표준 산업 분야의 응답자 현황을 다음과 같다.

<표 4-1> 산업별 응답자 현황

구분		비표준	자동차	철도	항공	총계
응답 기업 현황		15	11	11	13	50
기업 구분	대기업	1	6	4	4	15
	중소기업	12	5	7	8	22
	연구소	2	-	-	1	3
	소계	15	11	11	13	50
연구소보유기업		13	9	11	12	45
인증보유기업		7	3	6	8	24
수출기업		7	3	4	3	17

응답 현황으로 중소기업이 가장 많은 응답을 수행하였고, 연구소 보유 현황, 인증 보유 현황, 수출 기업 현황에 대한 기본 항목에 대한 설문을 수행하였다. 소프트웨어 안전성 실태 조사를 위해 해당 기업에서 안전성 관련 업무를 담당하거나 수행하는 대상자가 본 설문에 참여하였다.

비표준 분야의 경우 조선 해양 분야의 기업의 대부분은 품질 및 안전 인력을 보유하고 있고, 자동차의 경우 조사 기업이 대기업군은 안전성 관련 인력이 있으나 중소기업은 품질/안전성 인력이 거의 없는 것으로 나타났다.

조사 기업에 속한 기능 안전에 대한 국내외 표준 및 규제에 대한 조사 현황은 아래와 같이 조사되었다.

<표 4-2> 산업별 표준 현황

산업 분야	관련 표준
자동차	ISO 26262, TUV 라인란드 FSM, IATF 16949, A-SPICE, CMMi
철도	IEC 61508, EN50126, EN50128, EN50129, IEC62279, IEC62425, IEC62278, IEC 61511, IEC 62061
항공	RTCA DO-173, RTCA-DO-178/DO-178C, DO-254, MIL-HDBK-516C, AS 9100D, ISO 9100, MIL-STO-882E
비표준	IEC 61508, IEC와 항행통신장비표준, ISO 9001, ISO 140001, ASME, IMO SQA

※ 관련 표준에 대한 약어 정리는 부록 Ⅱ 참조

2. SW안전 관련 일반 실태 조사 분석

소프트웨어 안전 관련 일반 조사를 통해 산업군, 소프트웨어 수행 조직, 기업의 제품 소프트웨어 안전관련 일반 현황을 파악하고 분석하기 위한 조사이다. 산업군별로 소프트웨어안전 일반 현황, 소프트웨어안전 요구 제품 현황, 소프트웨어안전 제품 개발 프로세스 현황, 소프트웨어를 포함한 주요제품의 소프트웨어안전 관련 조사, 소프트웨어 개발 시 지원활동 관련 조사를 통해 소프트웨어 안전에 대한 일반 현황을 파악하고 분석한다.

1) SW안전 일반 현황

1. 귀사가 속한 산업군에 기능안전 관련하여 국내외 안전 표준/규제가 존재합니까? 존재한다면 표준/규제명을 적어 주시기 바랍니다. (복수 선택 가능)

<표 4-3> 국내외 안전 표준/규제에 대한 인식 현황 분석

구분	비표준		자동차		철도		항공		표준합계	
	건수	%	건수	%	건수	%	건수	%	건수	%
국제 표준/인증	7	47%	6	55%	9	82%	10	77%	25	71%
국내 법/제도	2	13%	0	0%	8	73%	4	31%	12	34%
있지만 명칭은 잘 정확히 모름	3	20%	1	9%	2	18%	1	8%	4	12%
잘모름	4	27%	3	27%	0	0	2	15%	5	14%

* 중복 응답, 응답기업수는 <표4-1> 참조, 백분율(%) 계산방식 = 응답수/각부분별 응답기업수

- 비표준 분야

비표준 분야의 국내외 안전 표준 및 규제는 IEC61508, ISO9001:2015, ISO14001, ASME, IMO SQA로 조사되었다. 안전 표준과 법제도에 대해 잘 모르는 경우가 27%로 조사되었다.

- 자동차

자동차 분야에서는 ISO 26262, IATF 16949, ISO26262, A-SPICE, CMMi, TUV 라인란드 FSM로 조사되었다. 안전 표준과 법제도에 대해 잘 모르는 경우가 27%로 조사되었다. 항공과 철도에 비해 자동차 분야는 아직 법제도 및 표준에 대한 지식이 부족하였다.

- 철도

철도 분야에서 국제 표준/인증으로는 IEC 61508, EN 50126, EN 50128, EN 50129, EN 20128, IEC 62278, IEC 62425, IEC 62279로 조사되었고, 국내 법/제도에는 철도 차량 기술기준, 철도 안전 형식 승인, 철도 안전법, 개발 보안 가이드로 조사되었다. 안전 표준과 법제도에 대해 조사에 참여한 기업 모두 알고 있는 것으로 조사되었다.

- 항공

항공 분야에서 국제 표준 및 인증은 DO-173, RTCA-DO-178, DO-178C, DO-254, MIL-HDBK-516C, AS9100D, IOS9100, DC187C, DO254, MIL-HDBK-516C, MIL-STO-882E로 조사되었고, 국내 법/제도에는 방위사업청 실무 매뉴얼인 무기체계 내장형 SW개발관리 지침, 군용 항공기 표준 감항 인증에 관한 고시 등을 적용하는 것으로 조사되었다. 안전 표준과 법제도에 대해 조사에 참여한 기업 모두 알고 있는 것으로 조사되었다. 국내 항공 개발 관련 기업은 국방 분야 관련 개발 업무가 대부분으로 국내 국방 관련 법제도에 대해 이해가 깊었다.

2.사업 발주/감독 시, 기능안전에 관련하여 개발요건에서 명시하고 있습니까?

① SW 기능안전 국제 표준을 명시적으로 이행 요구

② SW 기능안전 국제 표준에 맞는 자체 표준 제시 및 이행 요구

③ 현재는 요구하지 않으나 가까운 미래에 요구 예상

④ 요구하지 않음

<표 4-4> 발주/감독 시 기능안전 개발 요건 명시 현황 분석

구분	비표준		자동차		철도		항공		표준 계	
SW 기능안전 국제 표준을 명시적으로 이행 요구	1	7%	4	40%	7	64%	3	23%	14	42%
SW 기능안전 국제 표준에 맞는 자체 표준 제시 및 이행 요구	5	33%	1	10%	4	36%	7	54%	12	34%
현재는 요구하지 않으나 가까운 미래에 요구 예상	7	47%	4	40%	-	-	2	15%	6	17%
요구하지 않음	2	13%	1	10%	-	-	1	8%	2	6%

- 비표준 분야

비표준 분야에서는 “현재는 요구하지 않으나 가까운 미래에 요구 예상” 이 가장 많았고, “SW 기능 안전 국제 표준에 맞는 자체 표준 제시 및 이행 요구” 가 많은 응답을 자치하였다.

- 자동차

자동차 분야에서는 “SW 기능안전 국제 표준을 명시적으로 이행 요구” 와 “현재는 요구하지 않으나 가까운 미래에 요구 예상” 이 높은 응답을 하였다.

- 철도

철도 분야에서는 “SW 기능안전 국제 표준을 명시적으로 이행 요구” 가 많았고, 나머지는 “SW 기능안전 국제 표준에 맞는 자체 표준 제시 및 이행 요구” 로 응답하였다.

- 항공

항공 분야에서는 “SW 기능안전 국제 표준에 맞는 자체 표준 제시 및 이행 요구” 이 절반 이상이 응답하였고, “SW 기능안전 국제 표준을 명시적으로 이행 요구” 가 일분 응답을 하였다.

본 문항으로 발주처가 항공, 철도는 국제표준이나 자체표준을 이용하여 기능안전을 제시하도록 요구하고 있음을 사전 정보로 취득했다. 이에 따라 다른 조사 결과를 검토하고자 한다.

3. SW안전 프로세스의 적용 경험과 이해 수준에 대해 모두 체크 바랍니다.

	프로세스 항목	적용 경험	이해 수준
①	시스템 개발 시 위험 항목 관리	유 / 무	상 / 중 / 하
②	SW 위험원 관리 및 위험성 평가	유 / 무	상 / 중 / 하
③	요구사항 관리 관련 프로세스	유 / 무	상 / 중 / 하
④	개발관련 프로세스(분석/설계/구현/검증)	유 / 무	상 / 중 / 하
⑤	테스트 및 통합 관련 프로세스	유 / 무	상 / 중 / 하
⑥	품질 보증 프로세스	유 / 무	상 / 중 / 하
⑦	형상 관리 프로세스	유 / 무	상 / 중 / 하
⑧	자동화 된 툴 활용	유 / 무	상 / 중 / 하
⑨	안전 관리/인증 관련 프로세스	유 / 무	상 / 중 / 하

SW안전 프로세스 적용 경험에 대한 분석 결과는 아래와 같다.

<표 4-5> SW안전 프로세스 적용 경험 분석

(단위 : 건수, 백분율)

프로세스 항목	비표준		자동차		철도		항공		표준 합	
시스템 개발 시 위험 항목 관리	9	60%	6	55%	11	100%	11	85%	28	80%
SW 위험원 관리 및 위험성 평가	5	33%	6	55%	10	91%	10	77%	26	74%
요구사항 관리 관련 프로세스	12	80%	9	82%	11	100%	12	92%	32	91%
개발관련 프로세스	11	73%	10	91%	10	91%	12	92%	32	91%
테스트 및 통합 관련 프로세스	12	80%	9	82%	10	91%	12	92%	31	88%
품질 보증 프로세스	11	73%	9	82%	11	100%	11	85%	31	89%
형상 관리 프로세스	12	80%	8	73%	11	100%	11	85%	30	86%
자동화 된 툴 활용	7	47%	10	91%	11	100%	11	85%	32	91%
안전 관리/인증 관련 프로세스	6	40%	7	64%	10	91%	9	69%	26	74%

- 비표준 분야

비표준 분야에서는 형상관리 프로세스, 요구사항 관리 프로세스, 테스트 및 통합 관련 프로세스는 경험이 풍부한 것으로 나타났으나, 자동화 툴 활용, 안전 관리 및 인증 프로세스, 소프트웨어 위험원 관리 및 위험성 평가는 적용 경험이 낮은 것으로 나타났다.

- 자동차

자동차 분야에서는 요구사항 관리 프로세스, 개발 관련 프로세스, 테스트 및 통합 관련 프로세스, 품질 보증, 자동화 된 툴 활용이 높은 것으로 나타났다.

- 철도

철도 분야에서는 소프트웨어 전 분야의 프로세스에서 경험이 많은 것으로 조사되었고, 요구사항, 품질보증, 형상관리, 자동화 둘은 설문 조사 대상 모두가 사용하는 것으로 나타났다.

- 항공

항공 분야에서는 시스템 개발 시 위험 항목 관리, 요구사항 관리, 개발 관련 프로세스, 테스트 및 통합 관련 프로세스 등 전체적으로 철도 다음으로 적용 경험이 높은 것으로 나타났다.

산업 분야별 소프트웨어 안전 프로세스 적용 경험에 대한 분석으로 철도 분야가 가장 높게 나타났고, 항공, 자동차, 비표준 순으로 적용 경험이 분석되었다. 특히 철도 분야에서는 시스템 개발 시 위험 항목 관리, 요구사항 관리 관련 프로세스, 품질 보증 프로세스, 형상관리 프로세스는 모두 적용하는 것으로 분석되었다. 자동차, 비표준 분야는 일반적인 소프트웨어 개발프로세스는 비교적 잘 적용하고 있는 것으로 조사되었으나, 안전 제품 개발 시 가장 중요한 위험분석 및 안전 관리 측면이 부족한 것으로 나타났다. 특히 자동차 부분은 안전관리 프로세스 부분이 법제화되어 있지 않아, 중소기업 등은 안전 프로세스에 대한 이해가 낮은 것으로 나타났다.

SW안전 프로세스 적용 시 이해 수준에 대한 분석은 아래와 같다.

<표 4-6> SW안전 프로세스 적용 이해 수준 분석

(단위 :건수, 백분율)

프로세스 항목		비표준		자동차		철도		항공		표준 합	
시스템 개발 시 위험 항목 관리	상	2	14	3	33	7	36	6	46	13	37
	중	10	71	4	44	7	64	4	31	15	43
	하	2	14	2	22	0	0	3	23	5	14
SW 위험원 관리 및 위험성 평가	상	1	7	3	33	3	27	3	23	9	26
	중	6	43	4	44	8	73	6	46	18	51
	하	7	50	2	22	0	0	4	31	6	17
요구사항 관리 관련 프로세스	상	5	36	2	18	9	82	8	62	19	54
	중	8	57	7	64	2	18	3	23	12	34
	하	1	7	2	18	0	0	2	15	4	11
개발관련 프로세스	상	5	36	2	18	7	70	8	62	17	49
	중	7	50	8	73	3	30	4	31	15	43
	하	2	14	1	9	0	0	1	8	2	6
테스트 및 통합 관련 프로세스	상	3	21	3	27	6	60	8	62	17	49
	중	9	64	5	45	3	30	4	31	12	34

	하	2	14	3	27	1	10	1	8	2	14
품질 보증 프로세스	상	3	21	4	36	8	73	7	54	19	54
	중	8	57	4	36	3	27	4	31	11	31
	하	3	21	3	27	0	0	2	15	5	14
형상 관리 프로세스	상	5	26	3	27	8	73	8	62	19	54
	중	7	50	5	45	2	18	3	23	10	29
	하	2	14	3	27	1	9	2	15	6	17
자동화 된 툴 활용	상	3	21	1	9	4	36	7	54	12	34
	중	7	50	7	64	4	36	3	23	14	40
	하	4	29	3	27	3	27	3	23	9	26
안전 관리/인증 관련 프로세스	상	2	14	3	30	7	70	5	38	15	43
	중	4	29	3	30	2	20	6	46	11	31
	하	8	57	4	40	1	10	2	15	7	20

- 비표준 분야

비표준 분야에서는 SW 안전 프로세스 적용 이해 수준으로 전체 조사 프로세스 항목에 대한 수준이 “중” 이 가장 많은 것으로 분석되었다.

- 자동차

자동차 분야에서는 SW 안전 프로세스 적용 이해 수준으로 전체 조사 프로세스 항목에 대한 수준이 “중” 이 가장 많은 것으로 분석되었다.

- 철도

철도 분야에서는 SW 안전 프로세스 적용 이해 수준으로 요구사항 관리 프로세스, 개발 관련 프로세스, 테스트 프로세스, 품질 보증, 형상 관리 프로세스 등의 이해 수준이 “상” 으로 나타났다.

- 항공

항공 분야에서는 요구사항 프로세스, 개발 관련 프로세스, 테스트 프로세스, 품질 보증 프로세스, 형상 관리 프로세스 등의 이해 수준이 높은 것으로 분석되었다.

산업 분야별로 소프트웨어 안전 프로세스 적용 이해 수준에 대해서는 철도 분야와 항공 분야가 상대적으로 높은 수준을 갖고 있는 것으로 분석되었고, 특히 철도 분야는 소프트웨어 안전 프로세스 적용 경험과 이해 수준이 가장 높은 것으로 분석되었다.

2) SW안전 요구 제품 현황 실태 조사 분석

4. 개발하는 제품이나 시스템에서 소프트웨어 안전성 확보가 필요하다고 생각하십니까?

1. 전혀아니다 2. 아니다 3. 약간 아니다 4. 약간 그렇다 5. 그렇다 6.매우 그렇다

<표 4-7> 소프트웨어 안전 확보 필요성 분석

(단위 : 건수, 백분율)

산업분야	전혀아니다		아니다		약간아니다		약간그렇다		그렇다		매우그렇다	
비표준	**0**	**-**	**0**	**-**	**2**	**13**	**2**	**13**	**6**	**40**	**5**	**33**
자동차	0	-	0	-	0	-	2	18	6	55	3	27
철도	0	-	0	-	0	-	0	-	4	36	7	64
항공	0	-	0	-	0	-	1	8	2	14	10	77
표준합계	**0**	**-**	**0**	**-**	**0**	**-**	**3**	**9**	**12**	**34**	**20**	**57**

<표 4-8> 개발 제품(시스템)에서 소프트웨어 안전 확보 필요 이유 분석

(단위 :건수, 백분율)

구분	비표준		자동차		철도		항공		표준합계	
안전이 제품의 기본 기능	10	67	8	73	8	73	9	69	25	71
인증이나 인허가에 필요	7	47	4	36	6	55	6	46	16	46
제품 수출에 필요	3	20	1	9	5	45	3	23	9	26
고객이 원하는 경우	7	27	6	55	5	45	7	54	18	51

* 중복 응답, 응답기업수는 <표4-1> 참조, 백분율(%) 계산방식 = 응답수/각부분별 응답기업수

- 비표준 분야

비표준 분야에서는 안전성 확보의 필요성이 “그렇다”와 “매우 그렇다”가 73%로 조사되었다. 안전성 확보가 필요한 이유로 “안전이 제품의 기본적 기능”이고, “인증 및 인허가에 필요”가 높게 조사되었다. 기타의 이유로 사고 방지를 위해 안전이 필요하다고 조사되었다. 기타 의견으로 사고 방지에 안전이 필요하다고 생각한다고 조사되었다.

- 자동차

자동차 분야에서는 안전성 확보의 필요성이 “그렇다” 와 “매우 그렇다” 가 82%로 조사되었다. 안전성 확보가 필요한 이유로 “안전이 제품의 기본적 기능” 이고, “고객이 요구하거나” 나 “인증 및 인허가에 필요” 가 높게 조사되었다. 기타 의견으로 “SW오류로 인한 설비 연계 부분의 오류” 가 발생할 수 있기 때문에 소프트웨어 안전이 필요한 것으로 나타났다.

- 철도

철도 분야에서는 안전성 확보의 필요성이 “그렇다” 와 “매우 그렇다” 가 100%로 가장 높게 조사되었다. 안전성 확보가 필요한 이유로 “안전이 제품의 기본적 기능” 이고, “인증 및 인허가체 필요” 가 높게 조사되었다. 기타 의견으로 “대국민 서비스에 따른 안전성 확보가 필요” 하기 때문에 필요하다고 조사되었다.

- 항공

항공 분야에서는 안전성 확보의 필요성이 “그렇다” 와 “매우 그렇다” 가 92%로 가장 높게 조사되었다. 안전성 확보가 필요한 이유로 “안전이 제품의 기본적 기능” 이고, “고객이 원하는 경우” 가 높게 조사되었다. 기타 의견으로 관제 시스템의 안전성 확보에가 필요하다고 조사되었다.

산업 분야별로 소프트웨어 안전 확보 필요성 정도는 철도 분야가 “매우 그렇다”, “그렇다” 가 100%로 가장 높게 조사되었고, 항공, 철도, 자동차, 비표준 순으로 조사되었고, 개발 제품에서 소프트웨어가 필요한 이유에 대해서는 비표준, 자동차, 철도, 항공 분야 모두 “안전이 제품의 기본 기능” 이기 때문에 소프트웨어 안전 확보가 필요하다고 조사되어, **인증이나 직접적인 고객의 요구보다는 제품 자체의 품질을 높이기 위해 안전이 중요하다고 생각하는 것으로 조사되었다.**

3) SW안전 제품(시스템) 개발 프로세스 실태 조사 분석

5. 제품 개발 시 SW안전 개발 프로세스에 대해 어느 정도 수행한다고 생각하시니까?

1. 전혀아니다 2. 아니다 3. 약간 아니다 4. 약간 그렇다 5. 그렇다 6.매우 그렇다

<표 4-9> SW 안전 개발 프로세스에 대해 수행 정도 분석

(단위 :건수, 백분율)

산업분야	전혀아니다		아니다		약간아니다		약간그렇다		그렇다		매우그렇다	
비표준	2	13	2	13	2	13	4	27	3	20	2	13
자동차	1	9	1	9	1	9	4	36	2	18	2	18
철도	0	0	0	0	0	0	5	45	5	45	0	9
항공	0	0	2	15	3	23	2	15	5	38	1	8
표준합계	1	3	3	8	4	11	11	32	12	34	4	11

<표 4-10> SW 안전 개발 프로세스 참여 형태 분석

(단위 : 건수, 백분율)

구분	비표준		자동차		철도		항공		표준합계	
사내 독립된 안전관리 조직 및 안전관리 담당자(전문가) 참여	0	-	3	33	1	10	1	9	5	17
프로젝트 인원에 속하는 안전관리 담당자(개발 역할 겸임) 참여	11	79	1	11	3	30	7	64	11	35
프로젝트 인원에 속하는 안전관리 담당자 참여	1	7	0	-	6	60	2	18	8	26
사외 안전 관리 전문가 지원	0	-	1	11	0	-	0	-	1	4
기타	2	14	4	44	0	-	1	9	5	18

<표 4-11> SW 안전 개발 프로세스 적용이 미비한 이유 분석

(단위 : 건수, 백분율)

구분	비표준		자동차		철도		항공		표준합계	
진행 할 수 있는 가이드나 방법론이 없음	5	33	4	36	3	27	3	23	10	29
프로세스를 잘 알고 있는 인력 부족	9	60	6	55	4	36	8	62	18	51
적용 의지 부족 (의사결정자, 관리자, 개발자)	5	33	2	18	3	27	2	15	7	20
개발 프로세스에 적용하기 위한 환경 부족(일정, 비용, 인력, 지원)	6	40	7	64	6	55	9	69	22	63
잘 모름	1	7	2	18	0	0	0	0	2	6

* 중복 응답, 응답기업수는 <표4-1> 참조, 백분율(%) 계산방식 = 응답수/각부분별 응답기업수

- 비표준 분야

비표준 분야에서는 “그렇다, 매우 그렇다”가 33%로 조사되었고, “약간 그렇다”가 27%로 조사되었다.

소프트웨어 개발 프로세스는 “프로젝트 인원에 속하는 안전관리 담당자(안전관리, Verifier 등) 참여”의 형태가 가장 많은 것으로 조사되었다. 기타 의견으로는 “프로젝트 인원이 규정에 맞게 진행”하거나 “참여 형태를 구분하기 어렵다”라고 답변하였다.

소프트웨어 안전 개발 프로세스 적용이 잘 안 되는 이유로 “**프로세스를 잘 알고 있는 인력의 부족**”이 가장 많았고, “개발 프로세스에 적용하기 위한 환경이 부족”한 경우가 두 번째로 많이 나타났다. 기타 의견으로는 “아직은 필요성이 충분하지 않다”라고 답변한 의견도 있다.

- 자동차

자동차 분야에서는 “그렇다, 매우 그렇다”가 36%로 조사되었고, “약간 그렇다”가 36%로 조사되었다.

소프트웨어 개발 프로세스는 “독립된 안전 관리 조직 및 안전 관리 담당자 참여”와 “프로젝트 인원에 속한 안전 담당자 참여”형태가 가장 많은 것으로 조사되었다. 기타 의견으로 “개발 담당이 수행”, “제품별 복합적 운영(모두해당)”, 외주 개발 등의 의견이 조사되었다.

소프트웨어 안전 개발 프로세스 적용이 잘 안 되는 이유로 **“개발 프로세스에 적용하기 위한 환경이 부족”**이 가장 많았고, “프로세스를 잘 알고 있는 인력의 부족”한 경우가 다음으로 조사되었다. 기타 의견으로 “제품별 상황별로 다르게 적용되고 있다”라고 답변하였다.

- 철도

철도 분야에서는 “그렇다, 매우 그렇다”가 54%로 높게 조사되었고, “약간 그렇다”가 45%로 조사되었다.

소프트웨어 개발 프로세스는 “프로젝트 인원에 속하는 안전관리 담당자(개발자나 안전 담당자) 참여”의 형태가 가장 많은 것으로 조사되었다.

소프트웨어 안전 개발 프로세스 적용이 잘 안 되는 이유로 **“개발 프로세스에 적용하기 위한 환경이 부족”**, “프로세스를 잘 알고 있는 인력의 부족” 순서로 조사되었다.

- 항공

항공 분야에서는 “그렇다, 매우 그렇다”가 46%로 조사되었고, “약간 그렇다”가 15%로 조사되었다.

소프트웨어 개발 프로세스는 “프로젝트 인원에 속하는 안전관리 담당자(개발역할 겸임) 참여”의 형태가 가장 많은 것으로 조사되었다. 기타 의견으로 “안전 관리 인력 능력 부족하거나 인원수의 부족”하다는 의견이 조사되었다.

소프트웨어 안전 개발 프로세스 적용이 잘 안 되는 이유로 **“개발 프로세스에 적용하기 위한 환경이 부족”**이 가장 많았고, “ 프로세스를 잘 알고 있는 인력의 부족”한 경우가 다음으로 많이 조사되었다.

산업 분야를 종합해보면 SW 안전 개발 프로세스에 대해 수행 정도로 철도 분야가 “매우 그렇다, 그렇다”가 54%로 가장 높게 조사되었고, 참여 형태로는 프로젝트 인원에 속하는 안전관리 담당자(개발 역할 겸임) 참여나 프로젝트 인원에 속하는 안전관리 담당자(안전관리, verifier 등) 참여가 많은 것으로 조사되었다. 또한 소프트웨어 안전 개발 프로세스 적용이 **미비한 이유로는 산업 분야별로 공통적으로 “프로세스를 잘 알고 있는 인력이 부족”하거나 “개발 프로세스에 적용하기 위한 환경이 부족”한 것으로 조사되었다.**

6. SW안전 제품(시스템) 개발 시 안전계획(Functional Safety Plan)을 수립하고 있습니까?

1. 전혀아니다 2. 아니다 3. 약간 아니다 4. 약간 그렇다 5. 그렇다 6.매우 그렇다

<표 4-12> 제품 개발 시 안전계획을 수립 현황 분석

(단위 :건수, 백분율)

산업분야	전혀아니다		아니다		약간아니다		약간그렇다		그렇다		매우그렇다	
비표준	3	20	1	7	2	13	6	40	2	13	1	7
자동차	1	9	2	18	2	18	1	9	3	27	2	18
철도	0	0	1	9	0	0	2	18	6	55	2	18
항공	0	0	4	31	2	15	3	23	3	23	1	8
표준합계	1	3	7	19	4	11	6	17	12	35	5	14

<표 4-13> 안전 계획 수립이 어려운 이유 현황 분석

(단위 : 백분율)

구분	비표준		자동차		철도		항공		표준합계	
안전계획 수립에 대한 방법을 모름	4	27	3	27	3	27	5	38	11	31
안전계획에 대한 이해하고 있는 전문 인력 부족	11	73	7	64	6	55	8	62	21	60
안전계획을 수립하려고 하지만, 일정 부족	3	20	2	18	3	27	6	46	11	31
안전계획을 수립하려고 하지만, 예산 부족	6	40	4	36	3	27	6	46	13	37
안전계획을 수립하려고 하지만, 경영진 지원 부족	1	7	1	9	0	0	2	15	3	9
잘 모름	1	7	0	0	0	0	0	0	0	0

* 중복 응답, 응답기업수는 <표4-1> 참조, 백분율(%) 계산방식 = 응답수/각부분별 응답기업수

- 비표준 분야

비표준 분야에서는 “그렇다, 매우 그렇다”가 20%로 조사되었고, “약간 그렇다”가 40%로 조사되었다. 안전 계획 수립이 어려운 이유로는 “안전계획에 대한 이해하고 있는 전문 인력 부족”이 가장 많은 응답을 했고, “예산 부족”으로 어렵다고 하는 응답도 많이 조사되었다.

안전 계획 수립 시 포함되는 내용으로는 “고객 요구사항에 대한 기능 신뢰성”, “시험 방법 및 절차”, “안전 계획 정책, 개발 절차”, “기능 안전성 분석” 등이 포함되는 것으로 조사되었다.

- 자동차

자동차 분야에서는 “그렇다, 매우 그렇다”가 45%로 조사되었고, “약간 그렇다”가 9%로 조사되었다. 안전 계획 수립이 어려운 이유로는 비표준 분야와 유사하게 조사되었다.

안전 계획 수립 시 포함되는 내용으로는 “ISO26262 요구사항 전체 반영”, “차량 위치 추적과 제어를 담당하는 기능 안전”, “안전 검증 방안”, “안전과 관련된 SW프로세스 및 필요성”, “안전 활동, 형상 관리 등 지원 계획”, “안전전략 조직” 등이 포함되는 것으로 조사되었다.

- 철도

철도 분야에서는 “그렇다, 매우 그렇다”가 73%로 높게 조사되었고, “약간 그렇다”가 18%로 조사되었다. 안전 계획 수립이 어려운 이유로는 “안전계획에 대한 이해하고 있는 전문인력 부족”이 가장 많은 응답을 했고, “예산 부족”, “일정 부족” 등이 대체로 안전 계획 수립에 어려움을 주는 이유라고 답변하였다.

안전 계획 수립 시 포함되는 내용으로는 “ 안전관리조직(R&R), 안전관리프로세스, 산출물 등”, “철도 신호 시스템 기능 안전 부분”, “안전 위험원 분석 방법”, “신뢰성 및 안전성에 관련된 제반사항”, “안전성 관리계획서 작성 시 수명 주기 동안 발생할 위험원 저감 대책 등”, “RAMS안전성 분석과 SW주요 제어장치 부분에 대한 안전성” 등이 포함되는 것으로 조사되었다. 기타 의견으로 “고객의 요청 유무에 따라 안전 계획 수립 여부 결정”로 조사되었다.

- 항공

항공 분야에서는 “그렇다, 매우 그렇다”가 31%로 조사되었고, “약간 그렇다”가 23%로 조사되었다. 안전 계획 수립이 어려운 이유로는 철도 부분과 비슷한 양상을 보였다.

안전 계획 수립 시 포함되는 내용으로는 “SAE ARP4754에 다른 시스템 안전성 평가 계획”, “RTCA/DO-178”, “시스템 통합테스트 관련 스트레스 테스트”, “안전조직, 위험도 평가 절차, SW 안전성, 안전성 분석 및 평가”, “요구사항에서 산출물까지의 추적성 관리 방안, 신뢰성 시험 수준, 수행방법, 일정” 등이 포함되는 것으로 조사되었다.

산업 분야를 종합해보면 “그렇다, 매우 그렇다” 답변으로 철도가 가장 수준이 높았으며 자동차, 항공 순으로 높은 수준을 보였다. 비표준 분야는 타 분야에 비해 수준이 상대적으로 낮았다. 안전 계획 수립이 어려운 이유로는 공통적으로 **“안전 계획에 대한 이해하고 있는 전문 인력이 부족”** 한 것으로 조사되었다.

7. 위험 분석 수행 결과를 산출물로 관리하고 있습니까?

1. 전혀아니다 2. 아니다 3. 약간 아니다 4. 약간 그렇다 5. 그렇다 6.매우 그렇다

〈표 4-14〉 위험 분석 수행 결과 산출물 관리 현황 분석

(단위 : 건수, 백분율)

산업분야	전혀아니다		아니다		약간아니다		약간그렇다		그렇다		매우그렇다	
비표준	2	13	0	0	3	20	5	33	5	33	0	0
자동차	1	9	1	9	3	27	1	9	3	27	2	18
철도	0	0	0	0	0	0	1	9	9	82	1	9
항공	0	0	3	23	0	0	3	23	3	23	4	31
표준합계	1	3	4	11	3	9	5	14	15	44	7	20

〈표 4-15〉 위험 분석 수행 결과 산출물 관리가 어려운 현황 분석

(단위 : 건수, 백분율)

구분	비표준		자동차		철도		항공		표준합계	
산출물에 대한 작성 방법 모름	6	40	5	45	3	27	5	38	13	37
산출물 관리할 수 있는 인력 부족	9	60	3	27	6	55	6	46	15	43
산출물 관리를 위한 일정, 비용, 지원 부족	8	53	5	45	3	27	9	69	17	49
잘 모름 / 기타	1	7	5	45	0	0	0	0	5	14

* 중복 응답, 응답기업수는 〈표4-1〉 참조, 백분율(%) 계산방식 = 전체 응답수/각부분별 응답기업수

- 비표준 분야

비표준 분야에서는 “그렇다” 가 33%로 조사되었고, “약간 그렇다” 가 30%로 조사되었다. 위험 분석 수행 결과에 대한 산출물 관리가 미비한 이유로는 “산출물 관리할 수 있는 인력 부족”, “산출물 관리를 위한 일정, 비용, 지원이 부족” 한 것으로 조사되었다.

- 자동차

자동차 분야에서는 “그렇다, 매우 그렇다” 가 45%로 조사되었고, “약간 그렇다” 가 9%로 조사되었다. 위험 분석 수행 결과에 대한 산출물 관리가 미비한 이유로는 다른 분야와는 다르게 인력부족보다는 “산출물에 대한 작성 방법 모름”, “산출물 관리를 위한 일정,

비용, 지원이 부족” 으로 조사되었다.

- 철도

철도 분야에서는 “그렇다, 매우 그렇다” 가 91%로 높게 조사되었고, “약간 그렇다” 가 9%로 조사되었다. 위험 분석 수행 결과에 대한 산출물 관리가 미비한 이유로는 “산출물 관리할 수 있는 인력 부족” 이 가장 큰 것으로 조사되었다.

- 항공

항공 분야에서는 “그렇다, 매우 그렇다” 가 54%로 조사되었고, “약간 그렇다” 가 23%로 조사되었다. 위험 분석 수행 결과에 대한 산출물 관리가 미비한 이유로는 “산출물 관리를 위한 일정, 비용, 지원이 부족” 한 것이 가장 많은 것으로 조사되었다.

산업 분야를 종합해보면 “그렇다, 매우 그렇다” 답변으로 철도가 가장 수준이 높았으며 항공, 자동차 순으로 높은 수준을 보였다. 비표준 분야는 타 분야에 비해 수준이 상대적으로 낮았다. 공통적으로 인력 부족의 문제가 가장 큰 것으로 보인다.

위험 분석 수행 결과 산출물 명칭 및 관리 정보에 대한 조사는 아래와 같이 조사되었다.

<표 4-16> 위험 분석 수행 결과 산출물 명칭 및 관리 정보

분야	응답
비표준	• 위험 분석서, 위험 관리 요구사항, 표준 규정 내용 • 시험 성적서 – 기능 안전성 • 기능 안전 소스 코드 및 알고리즘
자동차	• 위험 분석 관련 : 시스템/HW/SW FMEA /FTA/ FMEDA 분석서 및 결과 • SW품질관리 보고서, 개발, 발생한 이슈 수집 • 통합테스트 계획서, 통합테스트 결과서 • safety goal/ ASIL/safe state등
철도	• 철도 신호 시스템 기능 중 안전에 직접 영향을 주는 위험원 분석 등 • 추적성관리표, 안전성 관리계획서, 검증확인 보고서, 요구사항 명세서등 • 위험원 리스트, 분석결과, (필요시) safety case
항공	• 기능위험평가보고서(FHA), 예비시스템 안전성평가 보고서(PSSA), 시스템 안전성 평가보고서(SSA), SSHA(서브시스템 위험요소분석), SRHA(시스템 요구도 위험요소 평가) • 위험 관리 대장

4) SW를 포함한 주요제품의 SW 안전관련 조사 분석

8. 제품(시스템)과 관련된 잠재적이고 비정상적인 동작을 신속하고 정확하게 식별하는데 효과적은 위험분석 방법은 무엇입니까?

1. 안전성 분석 기법 사용 2. 사내 위험분석 전문가의 경험
3.기존 보유한 시스템 정보를 사용 4. 외부 전문 컨설턴트 도움을 요청 5. 기타

<표 4-17> 위험 식별을 위해 효과적인 위험 분석 방법 현황 분석

(단위 :건수, 백분율)

구분	비표준		자동차		철도		항공		표준합계	
안전성 분석 기법 사용	2	15	1	13	4	44	3	25	8	27
사내 위험 분석 전문가의 경험	6	46	1	13	1	11	1	8	3	11
기존 보유한 시스템 정보를 사용	2	15	2	25	0	-	5	42	7	22
외부 전문 컨설턴트 도움 요청	3	23	2	25	4	44	3	25	9	31
기타	0	-	0	25	0	-	0	-	2	8

<표 4-18> SW 위험 분석 시 사용하는 기법 현황 분석

(단위 : 건수, 백분율)

구분	비표준		자동차		철도		항공		표준합계	
FTA	2	13	6	55	7	64	6	46	19	55
FMEA	4	27	7	64	9	82	7	54	23	66
HAZOP	1	7	2	18	7	64	2	15	11	32
STPA	1	7	1	9	1	9	0	0	2	6
기타	0	0	3	27	3	27	1	8	7	21
잘 모름(전혀 없음)	10	67	4	36	1	9	5	38	10	28

* 중복 응답, 응답기업수는 <표4-1> 참조, 백분율(%) 계산방식 = 응답수/각부분별 응답기업수

- 비표준 분야

비표준 분야에서는 위험 식별 분석 방법으로 , “사내 위험 분석 전문가의 경험”을 많이 하는 것으로 조사되었고, “외부 전문 컨설턴트 도움”이 두 번째로 많은 것으로 조사되었다. 위험 분석 기법으로는 FMEA(Failure mode and effects analysis), FTA(Fault Tree Analysis)를 많이 활용하고 있고, 잘 모르거나 기법을 활용하지 않는 다는 답변이 가장 많았다.

제품 중 소프트웨어의 위험 항목이 무엇인지에 대한 조사에서는 안전 지원 정보의 신뢰성, 시스템 오작동, 제어부 구동 문제, 기준 정보 오류, 보안 등을 들었으며, 해결 방법으로는 공인 시험, 코드 수정, 예외 사항 처리, 보안 강화 등으로 해결 한다는 의견을 제시하였다.

- 자동차

자동차 분야에서는 “기존 보유한 시스템 정보를 사용”, “외부 전문 컨설턴트 도움”이 가장 많은 것으로 조사되었다. 위험 분석 기법으로는 FMEA(Failure mode and effects analysis), FTA(Fault Tree Analysis)를 많이 활용하고 있고, 도구명으로는 “SOX2[38], IQ-FMEA[39], MKS[40]” 등을 활용하는 것으로 조사되었다. 도구 활용 방안으로 “변경관리, 형상관리, 검증, 시스템/하드웨어/소프트웨어 안전 분석”로 조사되었다. 기타 의견으로 FMEDA(Failure modes, effects, and diagnostic analysis)나 DFA(Design for Assembly)을 수행하는 것으로 조사되었다.

제품 중 소프트웨어의 위험 항목이 무엇인지에 대한 조사에서는 성애 제거 기능 오류, 외부 장비 신호 전달 오류, 급발진 등이 있는 것으로 조사되었고, 해결 방법으로는 발생 가능한 케이스를 사전에 시행하여 이상여부를 체크, 설비 연계 인터페이스, 통합테스트 수행, 시뮬레이터를 통한 검증, 안전 분석의 전 개발단계 적용을 통해 해결한다는 의견을 제시하였다.

- 철도

철도 분야에서는 “외부 전문 컨설턴트 도움”과 “안전성 분석 기법을 이용”하는 것으로 가장 많이 조사되었고, 위험 분석 기법으로는 FMEA, FTA, HAZOP 순으로 많이 활용하고 있고, 사용하는 도구로는 “정적분석, 동적 분석, LDRA TOOL[41], 코딩규칙 검사기[42], 동

38) EnCo(Engineers Consulting GmbH) Safety Office X2
39) 독일 APIS사의 고장 유형 및 영향 분석 도구
40) PTC Integrity Lifecycle Manager
41) Unit Testing Tool
42) 소스코드 기반 코딩 규칙 자동 검사 도구, 코딩 규칙으로 MISRA, ISO26262 등 국제 표준 외에도 발생 가능한 오류에 대한 규칙들을 추가로 제공

적 테스팅툴” 등을 활용하는 것으로 조사되었다. 도구 활용 방안으로 “커버리지 측정, 시험, 정적 분석, 동적 분석, 전 개발 단계에서 위험 분석 시 적용” 을 하는 것으로 조사되었다. 기타 의견으로 소프트웨어 단독으로 수행하기 않고 시스템 기능 단위로 위험도를 분석하거나 브레인스토밍을 통해 위험을 식별한다는 의견도 있었다.

제품 중 소프트웨어의 위험 항목이 무엇인지에 대한 조사에서는 신호장치 문제, 열차 속도 측정 및 이외 관련된 제어 문제, 데이타 무결성 부분 검증, 입력 및 출력 처리 문제, 코드 검증 등이 있는 것으로 조사되었고, 해결 방법으로는 독립 안전성 평가 수행, CRC(Cyclic Redundancy Checking) 체크, 시스템 이중화, 정적 분석 및 동적 분석 후 빌드 통합관리 등을 통해 해결한다는 의견을 제시하였다.

- 항공

항공 분야에서는 “기존 보유한 시스템 정보를 사용”, “외부 전문 컨설턴트 도움” 과 “안전성 분석 기법을 이용” 순으로 조사되었고, 위험 분석 기법으로는 FMEA, FTA, HAZOP 순으로 많이 활용하고 있고, 사용하는 도구로는 “Fault Tree Analysis(FTA) Software Tool, LDRA TOOL[43]” 등을 활용하는 것으로 조사되었다. 도구 활용 방안으로 “SW 신뢰성 시험, 정적시험(코딩규칙, 취약점점검), 안전성 분석” 등으로 활용하는 것으로 조사되었다. 기타 의견으로 소프트웨어 단위 보다는 시스템 단위로 위험 분석을 수행한다는 의견도 있었다.

제품 중 소프트웨어의 위험 항목이 무엇인지에 대한 조사에서는 항공기 비행 데이터 획득, 기능 실패에 의한 시스템 안전성 저해, 진단 신뢰도 하락 및 미검출, 배터리 폭발 위험성, 코딩 규칙 미준수, 소프트웨어 오류로 인한 MC(micro-controller) 고장, 결함으로 인한 비행 기능 상실 등이 있는 것으로 조사되었고, 해결 방법으로는 하드웨어 이중화 및 SW신뢰성 검사(동적 분석, 정적 분석), 소프트웨어 이중화(fault-tolerant)와 인증된 RTOS(실시간 운영체계) 사용, 하드웨어 및 소프트웨어 3중 안전장치 사용, MC(mission computer)를 듀얼 혹은 삼/이중화하여 MC(micro-controller) 고장에 대한 위험도 낮춤, 소프트웨어 신뢰성 시험 수행 및 감항 인증 프로세스 적용, 백업 기능 설계 등을 통해 해결한다는 의견을 제시하였다.

산업 분야를 종합해보면 위험 분석 현황으로 철도 분야에서는 안전성 분석 기법 사용과 외부 전문가의 요청을 많이 받고 있고, 비표준은 사내 위험 분석 전문가의 경험을 활용하고, 자동차 분야는 외부 전문가의 도움과 기존 보유한 시스템의 정보를 활용하고 있다고 조사되었다. 항공 분야는 기존 보유한 시스템 정보를 이용하는 경우가 많은 것으로 조사되었다. **위**

43) Unit Testing Tool

험 분석 적용 방식에 대해서는 분야별 공통점을 찾기가 어려웠으며, 각 분야별로 각 분야의 위험 분석 적용 방식의 장점을 공유하는 것이 필요할 것으로 보인다.

9. SW 제품의 안전 무결성등급(SIL, Safety Integrity Level)을 적용한다면 어떤 방법으로 안전 무결성등급을 정의하고 개발하십니까?

1. 발주처에서 제공 2. 기존 관행대로 선정 3.상황에 맞게 계산 및 산정
4. 기타 5. 모르겠음

<표 4-19> SW제품 안전 무결성 등급(SIL, Safety Integrity Level) 적용 방법 분석

(단위 :건수, 백분율)

구분	비표준		자동차		철도		항공		표준합계	
발주처에서 제공	7	47	5	33	5	50	5	42	15	42
기존 관행대로 선정	1	7	2	13	-	-	1	8	3	7
상황에 맞게 계산 및 산정	1	7	2	13	2	20	-	-	4	11
기타	-	-	3	20	3	30	3	25	9	25
모르겠음	6	40	3	20	-	-	2	25	6	15

- 비표준 분야

비표준 분야에서는 안전 무결성등급 정의는 “발주처 제공”, “모름” 순으로 조사되었다.

- 자동차

자동차 분야에서는 안전 무결성등급 정의는 “발주처 제공”, “기존 관행대로 선정”, “상황에 맞게 계산 및 산정” 순으로 조사되었다. 조향과 제동 장치 등은 ASIL D로 조사되었다.

- 철도

철도 분야에서는 안전 무결성등급 정의는 “발주처 제공”, “상황에 맞게 계산 및 산정” 순으로 조사되었다. 기타 의견으로 위험원 분석 뒤 내부 검토 회의를 안전무결성등급을 결정하는 것으로 조사되었다.

<표 4-20> 철도 분야 안전무결성 등급

등급	제품 등급
SIL 2	BRAKE 소프트웨어, Train Control & Monitoring System(TCMS)
SIL 4	신호 관련 소프트웨어, 전자연동 장치, 고속 철도용 디지털 송수신기, 열차 제어 시스템

- 항공

항공 분야에서는 안전 무결성 등급 정의는 “발주처 제공”, “기존 관행대로 선정” 순으로 조사되었다. 기타 의견으로 시스템 안전성평가 결과에 따름, 상황에 맞게 계산하여 산정, 민수항공기 및 시스템 개발 가이드(SAE ARP47S4A) 위험요소 평가를 통해 안전등급 정의한다는 의견도 조사되었다.

<표 4-21> 항공 분야 안전무결성 등급

등급	제품 등급
Level A	다기능 시현기
Level B	통제 시현 장치

산업 분야를 종합해보면 안전 무결성등급 적용 방법으로 비표준은 대부분 발주처에서 제공하거나 모른다고 답변하였다. **자동차 분야, 철도 분야와 항공 분야 모두 발주처에서 제공하는 경우가 가장 많은 경우로 조사되었다.**

5) SW 개발 시 지원활동 관련 현황 분석

10. 소프트웨어개발 형상항목을 변경 시 절차, 통제에 의한 수행한다고 생각하십니까?

1. 전혀아니다 2. 아니다 3. 약간 아니다 4. 약간 그렇다 5. 그렇다 6.매우 그렇다

<표 4-22> 형상 항목 변경 시 절차 및 통제 현황 분석

(단위 : 건수, 백분율)

산업분야	전혀아니다		아니다		약간아니다		약간그렇다		그렇다		매우그렇다	
비표준	2	13	0	0	1	7	7	47	4	27	1	7
자동차	0	0	0	0	2	18	4	36	5	45	0	0
철도	0	0	0	0	0	0	3	27	5	45	3	27
항공	0	0	2	15	0	0	6	46	3	23	2	15
표준합계	0	0	2	5	2	5	13	38	13	38	5	14

<표 4-23> 형상 통제 수행이 미비한 이유 분석

(단위 : 건수, 백분율)

구분	비표준		자동차		철도		항공		표준합	
형상 관리 인식 부족	6	40	6	55	4	62	8	62	18	51
형상 항목 식별 안됨	4	27	1	9	1	0	0	0	2	6
형상 관리에 대한 방법 모름	4	27	1	9	3	15	2	15	6	17
형상 관리를 위한 인력 부족	6	40	7	64	5	31	4	31	16	46
형상 관리를 위한 일정, 비용, 지원 부족	8	53	8	73	6	77	10	77	24	69
형상 관리를 툴 부족	3	20	1	9	3	8	1	8	5	14

* 중복 응답, 응답기업수는 <표4-1> 참조, 백분율(%) 계산방식 = 응답수/각부분별 응답기업수

<표 4-24> 소프트웨어 변경 요청 시 기록 관리되는 정보

(단위 : 건수, 백분율)

구분	비표준		자동차		철도		항공		표준합	
변경 요청자, 요청일, 내용	8	53	11	100	11	100	10	77	32	91
변경이 필요한 대상 산출물	5	33	5	45	9	82	10	77	24	69
변경이 필요한 대상 소스코드	10	67	9	82	10	91	7	54	26	74
변경에 소요되는 예상 공수	2	13	5	45	0	0	5	38	10	29
변경 처리 일정 계획	10	67	7	64	4	36	8	62	19	54
변경 승인 회의 결과	5	33	5	45	5	45	6	46	16	46
변경 처리 결과	6	40	9	82	9	82	9	69	27	77
변경 처리에 소요된 실제공수	3	20	6	55	1	9	0	0	7	20
변경 요청 기록 관리하지 않음	1	7	0	0	0	0	1	8	1	3

* 중복 응답, 응답기업수는 <표4-1> 참조, 백분율(%) 계산방식 = 응답수/각부분별 응답기업수

- 비표준 분야

비표준 분야에서는 “그렇다, 매우 그렇다”가 34%로 조사되었고, “약간 그렇다”가 47%로 조사되었다. 형상 통제 수행이 미비한 이유는 “형상 관리를 위한 일정, 비용, 지원 부족”, “형상 관리를 위한 인력 부족”, “형상 관리 인식 부족” 순으로 조사되었다.

변경 관리 요청시 기록 관리되는 정보로는 “변경 대상 소스 코드”와 “변경 처리 일정 계획” 순으로 관리되는 것으로 조사되었다.

- 자동차

자동차 분야에서는 “그렇다, 매우 그렇다”가 45%로 조사되었고, “약간 그렇다”가 36%로 조사되었다. 형상 통제 수행이 미비한 이유는 “형상 관리를 위한 일정, 비용, 지원 부족”, “형상 관리를 위한 인력 부족”, “형상 관리 인식 부족” 순으로 조사되었다.

변경 관리 요청시 기록 관리되는 정보로는 “변경 요청자, 변경 요청일, 변경 요청 내용”와 “변경되는 소스코드” 순으로 관리되는 것으로 조사되었다.

- 철도

철도 분야에서는 “그렇다, 매우 그렇다”가 73%로 높게 조사되었고, “약간 그렇다”가 27%로 조사되었다. 형상 통제 수행이 미비한 이유는 “형상 관리를 위한 일정, 비용, 지원 부족”, 형상 관리를 위한 인력 부족”, 형상 관리 인식 부족” 순으로 조사되었다.

변경 관리 요청시 기록 관리되는 정보로는 “변경 요청자, 변경 요청일, 변경 요청 내용”와 “변경되는 소스코드” 순으로 관리되는 것으로 조사되었다.

- 항공

항공 분야에서는 “그렇다, 매우 그렇다”가 38%로 조사되었고, “약간 그렇다”가 46%로 조사되었다. 형상 통제 수행이 미비한 이유는 “형상 관리를 위한 일정, 비용, 지원 부족”, 형상 관리 인식 부족” 순으로 조사되었다.

변경 관리 요청시 기록 관리되는 정보로는 “변경 요청자, 변경 요청일, 변경 요청 내용”와 “당 변경 요청 건에 의하여 변경이 필요한 대상 산출물” 순으로 관리되는 것으로 조사되었다.

산업 분야를 종합해보면 형상 항목 변경 프로세스 적용은 철도 분야에서 “그렇다, 매우 그렇다” 73%로 가장 많이 수행하고 있고, 항공 분야와 자동차 분야 순으로 조사되었고, 비표준은 상대적으로 낮게 조사되었다. 형상통제 수행이 미비한 이유로는 비표준 분야와 자동차, 철도, 항공 분야 모두 가장 큰 이유로 **“형상관리를 위한 일정, 비용, 지원이 부족”** 한 것으로 조사되었다. 두 번째로 항공 분야는 “형상관리 인식이 부족” 하다고 조사되었고, 비표준 분야, 철도, 자동차 분야에서는 **“형상관리를 위한 인력 부족”** 한 것으로 조사되었다.

<표 4-25> 분야별 형상 관리 도구

분야	형상 관리 도구
비표준	SVN, Git, Jira, Trello, CVD
자동차	Redmine, Git, Jira, CVS, SVN, PTC사의 Integrity, MKS, 체인지마이너, 제킨스, code beamer
철도	CODE INSPECTOR, CODE SONAR, CONTROLLER TESTER, SVN, 하베스트, Redmind, Git, SVN, 폴라이온
항공	SVN, Git, Vizend, Redmine, PVCS, Team Forg

3. 각 산업 분야별 소프트웨어 개발 프로세스 실태 분석

SW 안전개발 프로세스는 정의된 소프트웨어 프로세스, 개발방법론과 도구를 사용하여 소프트웨어를 개발하고 관리하는 데 필요한 과정을 의미하며 고객 안전관련 요구사항의 정의 및 변경관리, 소프트웨어 요구사항의 정의 및 명세, 아키텍처 설계 및 상세 설계, 구현, 테스트 등 개발 프로세스 실태를 분석합니다.

소프트웨어 안전 개발 프로세스 단계는 각 산업별로 동일하게 요구사항 명세, 아키텍처, 상세 설계, 구현, 테스트단계로 구분하여 조사하였다.

각 산업별로 총 5개의 단계와 15개의 공통 질문 및 분석결과로 구성되며 각 질문에는 분야별로 특화된 세부 질문들과 분석결과가 포함되어 있습니다. 거의 모든 질문들이 중복 응답을 허용하고 있으며, 계산식은 다음과 같다.

- 응답기업수는 <표4-1> 참조, 백분율(%) 계산방식 = 응답수/각부분별 응답기업수

1) 요구사항 명세 단계

1. 귀사는 소프트웨어 개발 시 안전 관련 요구사항을 충분히 관리하고 있습니까?

1. 전혀 아니다 2. 아니다 3. 약간 아니다 4. 약간 그렇다 5. 그렇다 6.매우 그렇다

<표 4-26> 안전 관련 요구사항 관리 현황 분석

(단위 : 건수, 백분율)

산업분야	전혀 아니다		아니다		약간 아니다		약간 그렇다		그렇다		매우 그렇다	
비표준	1	6.7	4	26.7	2	13.3	4	26.7	3	20.0	1	6.7
자동차	1	9.1	1	9.1	3	27.3	0	0.0	5	45.5	1	9.1
철도	0	0.0	0	0.0	1	9.1	3	27.3	3	27.3	4	36.4
항공	0	0	3	23.1	0	0	5	38.5	4	30.8	1	7.7
표준합	1	2.9	4	11.4	4	11.4	8	22.9	12	34.3	6	17.1

- 비표준 분야

소프트웨어 개발 시 안전 관련 요구사항 관리 수준에 대한 공통 질문에서, 비표준 분야에

서는 “그렇다, 매우 그렇다”가 26.7%, “약간 그렇다”가 26.7%로 조사되었다.

세부적으로 “요구사항 명세 시 수행하는 안전관련 활동”을 선택하라는 질문에 비표준 분야는 안전 활동에 대한 수행이 낮고, “도출된 요구사항은 모두 명세화되고, 변경 시 변경 사유를 기록”하는 활동이 가장 많은 것으로 조사되었다.

<표 4-27> 비표준 분야 요구사항 명세 시 수행하는 안전관련 활동

활동	건수	%
시스템의 안전 기능 구현을 위한 필요한 소프트웨어의 안전요구사항 충분히 할당	4	27%
안전관련 요구사항이 안전무결성등급을 만족할 수 있도록 상세히 기술	1	7%
하드웨어와 소프트웨어 간의 연동에 필요한 안전관련 제약 사항을 상세히 기술	5	33%
안전한 제품을 만드는데 필요한 소프트웨어 안전 기능 요구사항을 충분히 기술	6	40%
도출된 요구사항은 모두 명세화되고, 변경 시 변경 사유를 기록하고 있음	7	47%

“SW 기능안전 검증 계획과 관련된 활동”에 대한 분석으로는 비표준 분야에서는 SW기능안전 검증 계획과 관련한 활동이 매우 저조하게 나타났다.

<표 4-28> 비표준 분야 SW 기능안전 검증 계획과 관련된 활동

활동	건수	%
소프트웨어 수명주기 단계별 안전요구사항을 충족 확인하는 검증 계획 있음	2	13%
소프트웨어 안전 검증 계획에 검증시기, 수행인력, 검증환경/기법, 합격 기준 포함	4	27%
소프트웨어 안전 검증 계획에 대해 안전 관리자가 검토를 수행	3	20%

- 자동차 분야

소프트웨어 개발 시 안전 관련 요구사항 관리 수준에 대한 공통 질문에서, 자동차 분야의 경우 “그렇다, 매우 그렇다”가 54.6%로 조사되었으며 “약간 그렇다”고 응답한 경우는 없었다.

세부적으로 “요구사항 명세 시 수행하는 안전관련 활동”을 선택하라는 질문에 자동차 분야는 기술 안전 개념과 시스템 수준의 기술안전 요구사항 명세가 50%이하이고, “하드웨어-소프트웨어 인터페이스 명세서(ECU커넥터, MCU/IC, 메모리)” 작성 활동이 가장 많은 것으로 조사되었다.

<표 4-29> 자동차 분야 요구사항 명세 시 수행하는 안전관련 활동

활동	건수	%
기술 안전 개념 정의	5	45%
시스템 수준의 기술 안전 요구사항(TSR) 명세	5	45%
하드웨어-소프트웨어인터페이스명세서(ECU커넥터,MCU/IC, 메모리)	7	64%
소프트웨어 안전 요구사항 검증 계획 작성	6	55%
하드웨어-소프트웨어 인터페이스 검증 계획 작성	6	55%

"SW 기능안전 검증 계획과 관련된 활동" 에 대한 분석으로 자동차 분야는 "SW 안전 검증 계획에는 검증 시기, 수행인력, 검증환경/기법, 합격 기준 등이 포함" 하는 활동을 가장 많이 하는 것으로 조사되었다.

<표 4-30> 자동차 분야 SW 기능안전 검증 계획과 관련된 활동

활동	건수	%
SW 수명주기 단계별 안전 요구사항을 충족 확인하는 검증 계획이 있음	5	45%
SW안전 검증 계획에는 검증시기, 수행인력, 검증환경/기법, 합격기준 등 포함	6	55%
SW 안전 검증 계획에 대해 안전 관리자가 검토를 수행함	3	27%

- 철도 분야

소프트웨어 개발 시 안전 관련 요구사항 관리 수준에 대한 공통 질문에서 철도분야는 "그렇다, 매우 그렇다" 가 63.7%, "약간 그렇다" 가 27.3%로 조사되었다.

세부적으로 "요구사항 명세 시 수행하는 안전관련 활동" 을 선택하라는 질문에 철도 분야는 "요구사항을 완전하고 추적 가능하도록 기술" 하는 활동은 많이 하고, "주기적인 테스팅 기능 기술" 에 대한 안전 활동 수행은 잘 하지 않은 것으로 조사되었다.

<표 4-31> 철도 분야 요구사항 명세 시 수행하는 안전관련 활동

활동	건수	%
상위 문서를 준수하여 기술: 시스템 요구사항 명세서, 시스템 아키텍처 기술서 등	9	82%
요구사항을 완전하고 추적 가능하도록 기술	10	91%

소프트웨어 자체 고장과 오류에 대해 검지/보고하는 기능과 하드웨어 상태 확인 기능 기술	7	64%
시스템 안전 무결성 등급을 준수하기 위한 기능을 명확하게 식별	6	55%
시스템 안전 요구사항과 관련된 사항을 주기적으로 테스트하는 기능 기술	3	27%

철도 분야에서는 “소프트웨어 요구사항 적합성 검토를 위해 수행하는 활동”에 대한 분석으로 “요구사항 추적표를 통해 소프트웨어 요구사항이 시스템 요구사항으로부터 도출되었는지 검토”는 전반적으로 수행하는 것으로 조사되었다.

<표 4-32> 철도 분야 소프트웨어 요구사항 적합성 검토를 위해 수행하는 활동

활동	건수	%
소프트웨어 품질 보증 계획에 따라 작성자와 별도의 조직에서 검토	6	55%
요구사항 추적표를 통해 소프트웨어 요구사항이 시스템 요구사항으로부터 도출되었는지 검토	11	100%
소프트웨어 요구사항의 정확성, 완전성, 일관성, 구현 가능성, 표준 준수성, 테스트 용이성 검토	9	82%

추가적인 활동으로 “종합 소프트웨어 테스트를 명세하기 위해 수행하는 활동”에 대한 조사로 “테스트케이스는 실행순서, 입력값, 출력값, 완료기준 등을 포함하여 기술”한다는 응답이 가장 많았다.

<표 4-33> 철도 분야 종합 소프트웨어 테스트를 명세하기 위해 수행하는 활동

활동	건수	%
개발완료 소프트웨어의 테스트에 대해 목적, 테스트케이스, 환경, 완료기준 등을 기술	6	55%
테스트케이스는 실행순서, 입력값, 출력값, 완료기준 등을 포함하여 기술	9	82%

“요구사항 검증을 위해 수행하는 활동”에 대해서는 조사에서는 “소프트웨어 요구사항 명세서의 가독성과 추적성을 검증”한다는 활동이 가장 많은 것으로 조사되었다.

<표 4-34> 철도 분야 요구사항 검증을 위해 수행하는 활동

활동	건수	%
소프트웨어 요구사항 검증 보고서 작성	9	82%
소프트웨어 요구사항 명세서의 가독성과 추적성 검증	10	91%
테스트할 수 없는 요구사항의 정확한 커버리지를 증명하기 위한 추가 활동 정의	5	45%
소프트웨어 요구사항 명세서의 검증 결과 및 부적합 사항에 대한 해결책과 권고사항 명시	7	64%
안전성관점에서 하드웨어와 소프트웨어 간의 제약사항 고려	5	45%
소프트웨어 요구사항 명세서, 시스템 요구사항 명세서, 소프트웨어 품질보증계획서의 적합성 검증	7	64%

- 항공분야

항공 분야의 경우 “그렇다, 매우 그렇다” 가 45.5%, “약간 그렇다” 가 27.3%로 파악되었다. 세부적으로 “요구사항 명세 시 수행하는 안전관련 활동” 을 선택하라는 질문에 항공 분야는 “소프트웨어에 할당 된 시스템 기능 및 인터페이스의 안전 요구 사항을 분석” 한다는 응답이 가장 많았다.

<표 4-35> 항공 분야 요구사항 명세 시 수행하는 안전관련 활동

활동	건수	%
시스템 기능 및 인터페이스의 안전 요구사항을 분석함.	12	92%
시스템 위험 요소를 방지하기 위해 (소프트웨어에 할당 된) 시스템 안전 요구 사항을 다루는 상위 요구 사항을 정의함.	10	77%
상위 수준 요구사항은 소프트웨어 요구사항 표준을 준수함.	8	62%
상위 수준의 요구사항은 가능한 정량적으로 표현함. (적용할 수 있는 허용 오차를 사용)	5	38%
도출된 요구사항은 모두 명세화되고, 변경 시 변경 사유를 기록하고 있음	10	77%

또한, 항공분야의 검증과 관련된 활동으로 “안전관련 상위 수준 요구사항(high level requirements) 검증과 관련된 활동” 에 대한 질문으로 “시스템의 기능, 성능 및 안전관련 요구사항이 상위요구사항과 추적이 가능한지 검증” 한다는 응답이 가장 조사되었다.

<표 4-36> 항공 분야 안전관련 상위 수준 요구사항 검증과 관련된 활동

활동	건수	%
파생된(derived) 상위 수준 요구 사항은 시스템 안전성 평가 프로세스와 시스템 프로세스에 제공함.	5	38%
적절하지 않거나 부정확하다고 판단된 요구 사항은 시스템 프로세스 및 안전성 평가 프로세스로 피드백	4	31%
시스템 기능, 성능, 안전 관련 요구사항이 상위 수준 요구사항으로 충분히 도출되었는지 검증함.	8	62%
시스템의기능,성능 및 안전관련 요구사항이 상위 요구 사항과 추적이 가능한지 검증함.	**11**	**85%**

철도, 항공, 자동차 분야를 종합해 보면 안전 확보를 위한 요구사항 관련 활동에 가장 집중되는 활동은 상위 시스템의 안전 요구사항과 인터페이스를 분석하고 만족시키는 활동이며, 이를 검증하기 위한 활동으로는 요구사항 명세의 추적성 관리 및 검증이었다. 비표준 분야의 안전 요구사항 명세 부분은 아직 보완할 것이 많은 것으로 조사되었다.

“SW 개발 시 안전 관련 요구사항이 관리되고 있지 않는 이유” 에 대해 자유롭게 기술하도록 한 질문에는 분야별로 다음과 같은 의견들이 포함되었다.

<표 4-37> SW 개발 시 안전 관련 요구사항이 관리되고 있지 않는 이유 분석

분야	응답
비표준	• 인력 부족, 적은 인력 및 비용 부족, 투입 인력의 부족 • 관리 대장이 없음, 요구사항 부재, 안전관련 요구사항이 적거나 없음 • SW 공학에 대해 잘 모름, 필요성 인식 부족, 가이드라인이 없음
자동차	• 촉박한 일정으로 인해 • 인력 및 인식 부족 • 개발 요구사항과 안전 요구사항의 이원화된 관리
철도	• SIL이 할당된 SW만 안전관리 활동 수행 • 인력부족 • 비용과 시간이 드는 일(활동)이므로, 자체적으로 수행하기는 어려움
항공	• 예산, 일정 부족 • 고객의 요구사항이 없고 안전관련 개념 부족 • 체계수준의 안전관련 요구사항 분석 미비로 제대로 된 할당이 안됨 • SW개발프로세스가 명확히 정립되지 않음(세부적인 내용 부족)

2. 귀사는 요구사항 명세 단계에서 안전성 보증을 위한 기법을 충분히 적용하고 있다고 생각하십니까?

1. 전혀 아니다 2. 아니다 3. 약간 아니다 4. 약간 그렇다 5. 그렇다 6.매우 그렇다

<표 4-38> 요구사항 명세 단계에서 안전성 보증을 위한 기법 적용 현황 분석

(단위 :건수, 백분율)

산업분야	전혀 아니다		아니다		약간 아니다		약간 그렇다		그렇다		매우 그렇다	
비표준	2	13.3	4	26.7	2	13.3	5	33.3	2	13.3	0	0.0
자동차	1	9.1	1	9.1	1	9.1	5	45.5	3	27.3	-	0.0
철도	0	0.0	1	9.1	1	9.1	3	27.3	4	36.4	2	18.2
항공	0	0.0	4	30.8	0	0.0	6	46.2	2	15.4	1	7.7
표준합	1	2.9	6	17.1	2	5.7	14	40.0	9	25.7	3	8.6

- 비표준 분야

요구사항 명세 단계에서 안전성 보증을 위한 기법을 충분히 적용하고 있는지에 대한 공통 질문에서, 비표준 분야에서는 “그렇다” 가 13.3%, “약간 그렇다” 가 33.3%로 조사되었다. 세부적으로 “안전성 관점에서 요구사항 명세 시 사용하는 기법” 을 선택하라는 질문에 비표준 분야는 “비정형 표기법(그림, 다이어그램, 단순 도표 등)” 과 “시스템 안전 요구사항과 소프트웨어 안전 요구사항 간의 추적방법” 이라는 응답이 가장 많이 조사되었다. 기타 의견으로는 내부 규칙을 사용하거나 기법이 없는 경우도 조사되었다.

<표 4-39> 비표준 분야 안전성 관점에서 요구사항 명세 시 사용하는 기법

기법	건수	%
정형 표기법 (Zed, PVS, VMD등)	-	0%
준정형 표기법(제어흐름도, 데이터 흐름도, 상태천이도, UML등)	3	20%
비정형 표기법(그림, 다이어그램, 단순 도표 등)	5	33%
시스템 안전 요구사항과 소프트웨어 안전 요구사항 간의 추적방법	5	33%
안전 요구사항 관리 자동화 Tool 사용	2	13%

- 자동차 분야

요구사항 명세 단계에서 안전성 보증을 위한 기법을 충분히 적용하고 있는지에 대한 공통 질문에서, 자동차 분야의 경우 "그렇다"가 30%, "약간 그렇다"가 40%로 조사되었다.

세부적으로 "안전성 관점에서 요구사항 명세 시 사용하는 기법"을 선택하라는 질문에 자동차 분야는 "비정형 표기법(그림, 다이어그램, 단순 도표 등)"이 가장 많은 것으로 조사되었다. 기타 의견으로 요구사항 추적 매트릭스(SRTM) 기법을 사용하는 의견도 있었다.

<표 4-40> 자동차 분야 안전성 관점에서 요구사항 명세 시 사용하는 기법

기법	건수	%
자연어	7	64%
비정형 표기법(그림, 다이어그램, 단순 도표 등)	9	82%
준정형 표기법(제어흐름도, 데이터 흐름도, 상태천이도, UML등)	6	55%
정형 표기법 (Zed, PVS, VMD등)	0	0%
시스템 안전 요구사항과 SW안전 요구사항 간의 추적	5	45%
안전 요구사항 관리 자동화 Tool 사용	2	18%

- 철도 분야

요구사항 명세 단계에서 안전성 보증을 위한 기법을 충분히 적용하고 있는지에 대한 공통 질문에서 철도분야는 "그렇다, 매우 그렇다"가 54.6%, "약간 그렇다"가 27.3%로 조사되었다.

세부적으로 "안전성 관점에서 요구사항 명세 시 사용하는 기법"을 선택하라는 질문에 철도 분야는 "모델링(Modeling)"과 "구조적 방법론(Structured Methodology)"을 기법으로 활용하는 경우가 가장 많았다.

<표 4-41> 철도 분야 안전성 관점에서 요구사항 명세 시 사용하는 기법

기법	건수	%
자연어	3	27%
정형기법(수학적인 접근법에 기반)	1	9%
모델링 (Modeling)	8	73%
구조적 방법론 (Structured Methodology)	8	73%
결정 테이블 (Decision Tables)	4	36%

요구사항관리도구(예: Doors)	3	27%

철도 분야에서 추가적인 질문으로 “종합SW 테스트 시 사용하는 기법” 에 대한 질문에는 “성능 시험(스트레스 테스트, 반응 시간 및 메모리 제약사항, 성능 요구사항 등)” 에 대한 기법을 사용하는 경우가 73%로 많았다.

<표 4-42> 철도 분야 종합SW 테스트 시 사용하는 기법

기법	건수	%
성능 시험(스트레스 테스트, 반응 시간 및 메모리 제약사항, 성능 요구사항 등)	8	73%
기능 및 블랙박스 시험 (경계값 분석, 동치 클래스 및 입력 파티션 테스트)의 조합	7	64%

- 항공분야

요구사항 명세 단계에서 안전성 보증을 위한 기법을 충분히 적용하고 있는지에 대한 공통 질문에서 항공 분야의 경우 “그렇다, 매우 그렇다” 가 27.3%, “약간 그렇다” 가 36.4%로 파악되었다.

세부적으로 “안전성 관점에서 요구사항 명세 시 사용하는 기법” 을 선택하라는 질문에 항공 분야는 “준정형 표기법(제어흐름도, 데이터 흐름도, 상태천이도, UML등)” 과 “시스템 안전 요구사항과 SW안전 요구사항 간의 추적” 응답이 가장 많았다.

<표 4-43> 항공 분야 안전성 관점에서 요구사항 명세 시 사용하는 기법

기법	건수	%
자연어	5	38%
비정형 표기법(그림, 다이어그램, 단순 도표 등)	6	46%
준정형 표기법(제어흐름도, 데이터 흐름도, 상태천이도, UML등)	8	62%
정형 표기법 (Zed, PVS, VMD등)	0	0%
시스템 안전 요구사항과 SW안전 요구사항 간의 추적	8	62%
안전 요구사항 관리 자동화 Tool 사용	2	15%

산업 분야를 종합해보면 철도가 가장 수준이 높았으며 자동차, 항공 순으로 높은 수준을 보였다. 비표준 분야는 타 분야에 비해 수준이 상대적으로 낮았다. 평균적으로는 “그렇다, 매우 그렇다”가 31.3%, “약간 그렇다”가 34.2%로 나타났다. **사용하는 기법으로는 자동차의 경우는 비정형 표기법, 철도와 항공의 경우는 준정형 표기법을 사용하는 것으로 조사되었다.**

3. 요구사항 명세 단계의 산출물이 소프트웨어 설계에 필요한 정보를 충분히 제공하고 있습니까?

1. 전혀 아니다 2. 아니다 3. 약간 아니다 4. 약간 그렇다 5. 그렇다 6.매우 그렇다

〈표 4-44〉 요구사항 명세 단계의 산출물이 설계에 필요한 정보 현황 분석

(단위 :건수, 백분율)

산업분야	전혀 아니다		아니다		약간 아니다		약간 그렇다		그렇다		매우 그렇다	
비표준	1	7.1	1	7.1	-	0.0	8	57.1	4	28.6	-	0.0
자동차	0	0.0	-	0.0	4	36.4	3	27.3	3	27.3	1	9.1
철도	0	0.0	-	0.0	-	0.0	3	27.3	7	63.6	1	9.1
항공	-	0.0	1	7.7	2	15.4	4	30.8	5	38.5	1	7.7
표준합	**-**	**0.0**	**1**	**2.9**	**6**	**17.1**	**10**	**28.6**	**15**	**42.9**	**3**	**8.6**

- 비표준 분야

요구사항 명세 단계의 산출물이 소프트웨어 설계에 필요한 정보를 충분히 제공하고 있는지에 대한 공통 질문에서, 비표준 분야에서는 “그렇다”가 28.6%, “약간 그렇다”가 57.1%로 조사되었다. 세부적으로 “안전 요구사항 명세 관련 작성하는 산출물 목록”을 선택하라는 질문에 비표준 분야는 “소프트웨어 요구사항명세서(또는 요구사항정의서)”라는 응답이 가장 많이 조사되었다.

〈표 4-45〉 비표준 분야 안전 요구사항 명세 관련 작성하는 산출물 목록

산출물	건수	%
소프트웨어 요구사항명세서 (또는 요구사항정의서)	10	67%
소프트웨어 기능안전 검증계획서 (또는 소프트웨어 검증계획서)	4	27%
소프트웨어 요구사항 검증보고서	3	20%

- 자동차 분야

요구사항 명세 단계의 산출물이 소프트웨어 설계에 필요한 정보를 충분히 제공하고 있는지에 대한 공통 질문에서, 자동차 분야의 경우 “그렇다, 매우 그렇다”가 36.4%, “약간 그렇다”가 27.3%로 조사되었다. 세부적으로 “안전 요구사항 명세 관련 작성하는 산출물 목록”을 선택하라는 질문에 자동차 분야는 “하드웨어·소프트웨어 인터페이스 명세서(갱신)”, “소프트웨어 검증 계획서(갱신)”, “소프트웨어 검증 보고서” 작성이 가장 많은 것으로 조사되었다.

<표 4-46> 자동차 분야 안전 요구사항 명세 관련 작성하는 산출물 목록

산출물	건수	%
안전 계획	3	27%
모델링과 프로그램 언어를 위한 설계 및 코딩 지침	6	55%
소프트웨어 안전 요구사항 명세서	5	45%
하드웨어-소프트웨어 인터페이스 명세서(갱신)	7	64%
소프트웨어 검증 계획서(갱신)	7	64%
소프트웨어 검증 보고서	7	64%

- 철도 분야

요구사항 명세 단계의 산출물이 소프트웨어 설계에 필요한 정보를 충분히 제공하고 있는지에 대한 공통 질문에서 철도분야는 “그렇다, 매우 그렇다”가 72.7%, “약간 그렇다”가 27.3%로 조사되었다. 세부적으로 “안전 요구사항 명세 관련 작성하는 산출물 목록”을 선택하라는 질문에 철도 분야는 “소프트웨어 요구사항 명세서”라는 응답이 가장 많았다. 기타 의견으로 요구사항 명세 단계 산출물 중 “소프트웨어 시험 절차서에 입력, 종료 조건에 경계값을 명시하고 확인”하는 산출물도 작성하는 것으로 조사되었다.

<표 4-47> 철도 분야 안전 요구사항 명세 관련 작성하는 산출물 목록

산출물	건수	%
소프트웨어 요구사항 명세서	8	73%
종합 소프트웨어 테스트 명세서	7	64%
소프트웨어 요구사항 검증 보고서	1	9%

- 항공분야

요구사항 명세 단계의 산출물이 소프트웨어 설계에 필요한 정보를 충분히 제공하고 있는지에 대한 공통 질문에서 항공 분야의 경우 “그렇다, 매우 그렇다”가 54.6%, “약간 그렇다”가 18.2%로 파악되었다. 세부적으로 “안전 요구사항 정의 산출물에 포함되는 항목”을 선택하라는 질문에 항공 분야는 “고장(Failure) 탐지 및 안전 모니터링 요구사항 ”이라는 응답이 가장 많았다.

<표 4-48> 항공 분야 안전 요구사항 정의 산출물에 포함되는 항목

산출물에 포함된 항목	건수	%
안전 관련 요구 사항 및 잠재적 장애 조건	4	31%
고장(Failure) 탐지 및 안전 모니터링 요구 사항	11	85%
소프트웨어에 할당 된 파티션 요구 사항	3	23%
파티션 된 소프트웨어 구성 요소가 서로 상호 작용하는 방법 및 각 파티션의 소프트웨어 수준	4	31%

산업 분야를 종합해보면 철도가 가장 수준이 높았으며 항공, 자동차 순으로 높은 수준을 보였다. 비표준 분야는 타 분야에 비해 수준이 상대적으로 낮았다. 평균적으로는 “그렇다, 매우 그렇다”가 48%, “약간 그렇다”가 32.5%로 나타났다.

2) 아키텍처 설계 단계

4. 귀사는 아키텍처 설계과정에서 요구 사항을 충분히 반영한 설계가 이루어지고 있다고 생각하십니까?

1. 전혀 아니다 2. 아니다 3. 약간 아니다 4. 약간 그렇다 5. 그렇다 6.매우 그렇다

<표 4-49> 요구사항이 아키텍처 설계에 어느 정보 반영되는지 현황 분석

(단위 :건수, 백분율)

산업분야	전혀 아니다		아니다		약간 아니다		약간 그렇다		그렇다		매우 그렇다	
비표준	1	6.7	1	6.7	-	0.0	7	46.7	6	40.0	-	0.0
자동차	1	9.1	1	9.1	3	27.3	2	18.2	4	36.4	-	0.0
철도	0	0.0	1	9.1	1	9.1	1	9.1	7	63.6	1	9.1
항공	-	0.0	-	0.0	1	7.7	2	15.4	8	61.5	2	15.4
표준합	1	2.9	2	5.7	5	14.3	5	14.3	19	54.3	3	8.6

- 비표준 분야

아키텍처 설계과정에서 요구 사항을 충분히 반영한 설계가 이루어지고 있는지에 대한 공통 질문에서, 비표준 분야에서는 “그렇다” 가 40%, “약간 그렇다” 가 46.7%로 조사되었다.

세부적으로 “아키텍처 설계를 위해 수행하는 활동” 을 선택하라는 질문에 비표준 분야는 “소프트웨어 요구사항 명세를 기반으로 소프트웨어 아키텍처 명세를 작성” 과 “소프트웨어 컴포넌트를 식별하여 명세화 함” 이라는 응답이 가장 많이 조사되었다.

<표 4-50> 비표준 분야 아키텍처 설계를 위해 수행하는 활동

활동	건수	%
소프트웨어 요구사항 명세를 기반으로 소프트웨어 아키텍처 명세를 작성	11	73%
모든 하드웨어/소프트웨어 상호 작용을 식별하고 분석하여 상세하게 작성	6	40%
소프트웨어 컴포넌트를 식별하여 명세화 함	10	67%
소프트웨어와 응용 데이터 / 알고리즘 간의 상세한 인터페이스를 명시함	4	27%

“아키텍처 설계를 위해 수행하는 안전관련 활동” 에 대한 분석으로는 비표준 분야에서는 “안전기능 설계・구현 시 하드웨어・시스템 엔지니어 협의” 이라는 응답이 가장 많았다.

<표 4-51> 비표준 분야 아키텍처 설계를 위해 수행하는 안전관련 활동

활동	건수	%
안전기능 설계/구현 시 하드웨어/시스템 엔지니어 협의	8	53%
시험 가능성 및 변경 용이성을 고려한 설계	8	53%
소프트웨어 안전/비안전 기능간의 독립성 확보	4	27%
안전 무결성 보증을 위한 제어흐름, 데이터 흐름 모니터링 가능 여부	6	40%

- 자동차 분야

아키텍처 설계과정에서 요구 사항을 충분히 반영한 설계가 이루어지고 있는지에 대한 공통 질문에서, 자동차 분야의 경우 “그렇다” 가 36.4%, “약간 그렇다” 가 18.2%로 조사되었다.

세부적으로 “아키텍처 설계를 위해 수행하는 활동” 을 선택하라는 질문에 자동차 분야는 “결함이나 고장의 검출을 위한 추가 요구사항 식별” 이 가장 많은 것으로 조사되었다.

<표 4-52> 자동차 분야 아키텍처 설계를 위해 수행하는 활동

활동	건수	%
안전 목표 및 안전 개념의 타당성 확인	4	36%
안전 개념 및 안전 요구사항 검증	4	36%
안전 목표나 안전 요구사항의 위배를 야기할 수 있는 결함 및 고장 조건과 원인 식별	5	45%
결함이나 고장의 검출을 위한 추가 요구사항 식별	9	82%
검출된 결함이나 고장에 대한 필수(동작/수단) 결정	5	45%
안전 관련 차량 시험에 포함될 안전 목표, 안전요구사항 검증을 위한 추가 요구사항 식별	3	27%

“아키텍처 설계를 위해 수행하는 안전관련 활동” 에 대한 질문에서 자동차 분야는 “제한된 크기의 소프트웨어 컴포넌트・인터페이스 식별” 이라는 응답이 가장 많았다.

<표 4-53> 자동차 분야 아키텍처 설계를 위해 수행하는 안전관련 활동

활동	건수	%
소프트웨어 컴포넌트 유닛까지 식별되는 계층적 구조로 분할 식별	4	36%
제한된 크기의 소프트웨어 컴포넌트/인터페이스 식별	6	55%
SW 안전요구사항에 대한 SW 컴포넌트 할당	5	45%
컴포넌트 할당된 요구사항에 대한 ASIL 부합 설계	2	18%
각 소프트웨어 컴포넌트 내 높은 응집도(cohesion) 기반 설계	5	45%
소프트웨어 컴포넌트 사이 제한된 결합도(coupling) 기반 설계	4	36%
적합한 스케줄링 특성 식별	4	36%
소프트웨어 컴포넌트 상세설계	3	27%
소프트웨어 컴포넌트 정적 설계 (sequence diagram, HW SW 인터페이스 명세)	5	45%
소프트웨어 컴포넌트 동적 설계 (sequence diagram, task structure 정의)	5	45%

“안전 무결성을 유지할 명확한 아키텍처 표현법 보유와 보유한 표현법을 기술하시오” 라는 질문에 대한 질문에서 자동차 분야는 “UML Class Diagram 사용” 과 “ASIL이 할당되는 소프트웨어 컴포넌트의 경우 ASIL이 표기” 라는 응답이 가장 많았다.

<표 4-54> 안전무결성을 유지할 명확한 아키텍처 표현법 보유와 보유한 표현법

표현법	건수	%
UML Class Diagram 사용	5	45%
소프트웨어 컴포넌트 리스트에 정의된 전체 소프트웨어 컴포넌트 표기	4	36%
Middle ware, device driver와 같은 하드웨어 관련 컴포넌트들이 같이 표기	3	27%
ASIL이 할당되는 소프트웨어 컴포넌트의 경우 ASIL이 표기	5	45%

- 철도 분야

아키텍처 설계과정에서 요구 사항을 충분히 반영한 설계가 이루어지고 있는지에 대한 공통 질문에서 철도분야는 “그렇다, 매우 그렇다” 가 72.7%, “약간 그렇다” 가 9.1%로 조사되었다.

세부적으로 “소프트웨어 아키텍처 명세 시 수행하는 활동” 을 선택하라는 질문에 철도분야는 “소프트웨어 요구사항 명세를 기반으로 소프트웨어 아키텍처 명세를 작성” 한다는 응답이 가장 많았다. 기타 의견으로 “고객 요구사항에 명시되어 있는 경우만 적용” 한다는 의견도 조사되었다.

<표 4-55> 철도 분야 소프트웨어 아키텍처 명세 시 수행하는 활동

활동	건수	%
소프트웨어 요구사항 명세를 기반으로 소프트웨어 아키텍처 명세 작성	11	100%
모든 하드웨어/소프트웨어 상호 작용을 식별하고 분석하여 상세하게 작성	9	82%
소프트웨어 컴포넌트를 식별하고, 컴포넌트별 신규/기존재 여부, 기존 컴포넌트의 검증 수행	8	73%
소프트웨어 컴포넌트를 형상 관리 시스템 내에서 독립적으로 버전 관리	4	36%
표준에 따라 개발하고 검증된 기존 소프트웨어 컴포넌트 재사용	5	45%
소프트웨어와 응용 데이터 / 알고리즘 간의 상세한 인터페이스 명시	6	55%
소프트웨어 안전 무결성 등급에 따른 소프트웨어 개발 전략 수립	4	36%

추가적인 활동으로 “소프트웨어 인터페이스 명세를 작성하기 위해 수행하는 활동”에 대한 조사로 “소프트웨어 컴포넌트와 전체 소프트웨어의 경계에 대한 인터페이스를 기술”한다는 응답이 가장 많았다.

<표 4-56> 소프트웨어 인터페이스 명세를 작성하기 위해 수행하는 활동

활동	건수	%
소프트웨어 컴포넌트와 전체 소프트웨어의 경계에 대한 인터페이스 기술	8	73%
인터페이스 명세는 사전/사후조건, 경계값 및 초과시 동작, 기능간 동기화 메커니즘 등을 포함	6	55%

“소프트웨어 아키텍처 명세 작성 시 수행하는 활동”에 대한 질문에서는 “소프트웨어 안전 무결성 등급의 요구사항이 구현 가능한지 분석”과 “기존 소프트웨어 재사용을 위해, 요구사항, 가정, 다른 SW와의 인터페이스 식별 및 문서화”한다는 응답이 가장 많은 것으로 조사되었다. 기타 수행 활동으로는 “인터페이스 용어에 대한 재정의”을 수행한다고 조사되었다.

<표 4-57> 철도 분야 소프트웨어 아키텍처 명세 작성 시, 수행하는 활동

활동	건수	%
소프트웨어 안전 무결성 등급의 요구사항이 구현 가능한지 분석	7	64%
소프트웨어 아키텍처는 애플리케이션의 안전 부분의 크기와 복잡성을 최소화함	5	45%
기존 소프트웨어 재사용을 위해, 요구사항, 가정, 다른 SW와의 인터페이스 식별 및 문서화	7	64%
소프트웨어 검증 프로세스에 기존 소프트웨어를 포함하여 수행한다.	3	27%
장애 회피와 장애 대응 전략 간의 균형을 이루어 장애 처리 수단 수립	4	36%
선정한 기법, 대책 및 도구가 요구 안전 무결성 등급의 요구사항 명세를 만족함을 증명	2	18%

“소프트웨어 안전 무결성 등급이 SIL 3 또는 SIL 4의 경우의 아키텍처 설계 시 수행하는 활동”에 대한 질문에서는 “기존 소프트웨어의 오류 감지 시, 시스템이 이러한 오류로부터 보호됨을 검증 및 확인”이라는 응답이 가장 많은 것으로 조사되었다.

〈표 4-58〉 SIL 3 또는 SIL 4의 경우의 아키텍처 설계 시 수행하는 활동

활동	건수	%
기존 소프트웨어의 예상 가능한 실패 및 전체 소프트웨어에 대한 영향 분석	7	64%
기존 소프트웨어의 고장 탐지 및 고장으로부터 시스템을 보호 전략 정의	5	45%
기존 소프트웨어가 할당된 요구사항을 충족함을 검증 및 확인	7	64%
기존 소프트웨어의 오류 감지시, 시스템이 이러한 오류로부터 보호됨을 검증	3	27%
기존 소프트웨어의 환경에 대한 가정이 성취되었음을 검증 및 확인	4	36%

"소프트웨어 설계 명세 작성 시, 안전을 고려하여 수행하는 활동"에 대한 질문에서는 "상위문서를 기반으로 소프트웨어 설계 명세를 작성"과 "컴포넌트와 외부와의 인터페이스, 컴포넌트 간 인터페이스, 데이터구조를 명시"한다는 응답이 가장 많은 것으로 조사되었다.

〈표 4-59〉 소프트웨어 설계 명세 작성 시, 안전을 고려하여 수행하는 활동

활동	건수	%
상위문서를 기반으로 소프트웨어 설계 명세를 작성한다.	11	100%
컴포넌트와 외부와의 인터페이스, 컴포넌트간 인터페이스, 데이터구조 명시	11	100%
컴포넌트에 대한 요구사항 할당 및 추적, 주요 알고리즘과 처리 순서, 오류 보고 메커니즘 명시	5	45%
코딩 표준 개발 및 명시하고, 이에 따라 개발	10	91%
소프트웨어 아키텍처와 안전 무결성 등급에 대한 소프트웨어 컴포넌트의 추적성 기술	5	45%
소프트웨어 안전 무결성 등급 요구 범위에서 코딩 표준에 대한 근거 제시	3	27%

"소프트웨어 통합 테스트 명세를 작성하기 위해 수행하는 활동"에 대한 질문에서는 "상위문서에 기초하여, 소프트웨어 통합 테스트 명세를 작성"한다는 응답이 가장 많은 것으로 조사되었다.

〈표 4-60〉 소프트웨어 통합 테스트 명세를 작성하기 위해 수행하는 활동

활동	건수	%
상위문서에 기초하여, 소프트웨어 통합 테스트 명세 작성	11	100%
소프트웨어 통합 테스트 명세 작성 시 다음의 사항 포함: 테스트 목적, 테스트케이스, 테스트 유형, 환경, 완료 판정 기준, 커버리지 기준, 책임	10	91%

“소프트웨어/하드웨어 통합 테스트 명세를 위해 수행하는 활동”에 대한 질문에서는 “상위문서에 기초하여 소프트웨어/하드웨어 통합 테스트 명세서 작성”한다는 응답이 가장 많은 것으로 조사되었다. 기타 의견으로 “장비 용량 시험을 진행”하거나 “소프트웨어/하드웨어통합 명세는 별도 작성”하지 않는 경우도 있다고 조사되었다.

〈표 4-61〉 소프트웨어/하드웨어 통합 테스트 명세를 위해 수행하는 활동

활동	건수	%
상위문서에 기초하여 소프트웨어/하드웨어 통합 테스트 명세서 작성	10	91%
소프트웨어/하드웨어 통합 테스트 명세는 하드웨어와 소프트웨어 요구를 테스트할 수 있도록 작성	8	73%
소프트웨어/하드웨어 통합 테스트 명세에 다음의 사항 포함: 테스트케이스와 테스트 데이터, 수행할 테스트 유형, 도구, 지원 소프트웨어, 설정을 포함한 테스트 환경, 테스트 완료 판정 기준	8	73%

“소프트웨어 아키텍처 및 설계 검증을 위해 수행하는 활동”에 대한 질문에서는 “상위문서에 기초하여 소프트웨어 아키텍처 및 설계검증 보고서를 작성”한다는 응답이 가장 많은 것으로 조사되었다. 기타 의견으로 “ 요구사항 추적이 어렵고”, “시간과 인력의 부족”으로 수행이 어려운 경우도 있다고 조사되었다.

〈표 4-62〉 소프트웨어 아키텍처 및 설계 검증을 위해 수행하는 활동

활동	건수	%
상위 문서에 기초하여 소프트웨어 아키텍처 및 설계 검증 보고서 작성	10	91%
소프트웨어 아키텍처/인터페이스/설계 명세는 완전성, 일관성, 적합성, 가독성, 추적성 보장	8	73%
소프트웨어 통합 및 소프트웨어/하드웨어 통합 테스트 명세는 가독성, 추적성 보장.	8	73%

- 항공분야

아키텍처 설계과정에서 요구 사항을 충분히 반영한 설계가 이루어지고 있는지에 대한 공통 질문에서 항공 분야의 경우 “그렇다, 매우 그렇다”가 72.7%, “약간 그렇다”가 18.2%로 파악되었다.

세부적으로 “아키텍처 설계를 위해 수행하는 활동”을 선택하라는 질문에 항공 분야는

“소프트웨어 요구사항을 기반으로 소프트웨어 아키텍처 정의”와 “소프트웨어 컴포넌트를 식별하여 명세화” 한다는 응답이 가장 많았다.

〈표 4-63〉 항공 분야 아키텍처 설계를 위해 수행하는 활동

활동	건수	%
소프트웨어 요구사항을 기반으로 소프트웨어 아키텍처 정의	11	85%
모든 하드웨어/소프트웨어 상호 작용을 식별하고 분석	10	77%
소프트웨어 컴포넌트를 식별하여 명세화	11	85%
소프트웨어와 응용 데이터 / 알고리즘 간의 상세한 인터페이스를 정의	9	69%

“아키텍처 설계를 위해 수행하는 안전관련 활동”에 대한 질문에서는 “소프트웨어 아키텍처가 상위 수준 요구 사항을 충족시키는지 검증” 한다는 응답이 가장 조사되었다.

〈표 4-64〉 항공 분야 아키텍처 설계를 위해 수행하는 안전관련 활동

활동	건수	%
소프트웨어 아키텍처가 상위 수준 요구 사항을 충족시키는지 검증	12	92%
소프트웨어 아키텍처의 구성 요소 간에 관계가 올바르다는 것을 검증	11	85%
소프트웨어 아키텍처와 대상 컴퓨터의 하드웨어 / 소프트웨어 간에 충돌이 없음을 검증 (특히 초기화, 비동기 작업, 동기화, 인터럽트)	7	54%
파티셔닝 위반이 없음을 검증	4	31%

아키텍처 설계 활동에서 가장 중요하여 반드시 수행되는 활동은 소프트웨어 요구사항 명세를 기반으로 소프트웨어 아키텍처 명세를 작성하는 활동으로 조사되었다. 그 다음 중요시 되는 활동이 하드웨어와 소프트웨어 상호 작용을 식별하고 분석하는 활동이다. 안전 확보를 위해 수행된 활동은 소프트웨어 안전 무결성 등급의 요구사항이 구현 가능한지 분석하는 활동이다. 자동차 분야에 특화된 활동은 제한된 크기의 소프트웨어 컴포넌트와 인터페이스 식별이다.

“아키텍처 설계 시 요구사항을 충분히 반영한 설계가 이루어지지 않은 이유”로는 “요구사항의 불명확, 인력/비용/인력 부족, SW개발 시 각 단계별 업무의 세분화된 프로세스 미흡을 이유로 들었다.

<표 4-65> 요구사항이 아키텍처에 잘 반영되지 않은 이유 분석

분야	응답
비표준	• SW공학에 대해 잘 모르고, 추적관리 프로세스 방법론 부재 • 시간부족
자동차	• 일정, 비용, 인력 부족 • 인식부족, 외주개발로 진행 • 설계기간이 짧거나 요구사항이 불명확하며, 아키텍처 개념 이해 부족
철도	• 요청사항 추적의 어려움 • 시간과 인력의 부족
항공	• 요구사항 불명확 • 아키텍처 설계를 상세하게 할 수 있는 리소스(인력, 비용, 일정) 부족 • SW개발 시 각 단계별 업무의 세분화된 프로세스 미흡에 의해 놓치고 있음

5. 귀사는 아키텍처 설계 단계에서 안전성 보증을 위한 기법을 충분히 적용하고 있다고 생각하십니까?

1. 전혀 아니다 2. 아니다 3. 약간 아니다 4. 약간 그렇다 5. 그렇다 6.매우 그렇다

<표 4-66> 아키텍처 설계에서 안전성 보증 기법을 어느 정도 적용하는지 현황 분석

(단위 : 건수, 백분율)

산업분야	전혀 아니다		아니다		약간 아니다		약간 그렇다		그렇다		매우 그렇다	
비표준	1	6.7	5	33.3	1	6.7	7	46.7	1	6.7	-	0.0
자동차	1	10.0	3	30.0	1	10.0	2	20.0	3	30.0	-	0.0
철도	0	0.0	-	0.0	2	18.2	2	18.2	5	45.5	2	18.2
항공	-	0.0	4	30.8	1	7.7	5	38.5	2	15.4	1	7.7
표준합계	1	2.9	7	20.6	4	11.8	9	26.5	10	29.4	3	8.8

- 비표준 분야

아키텍처 설계 단계에서 안전성 보증을 위한 기법을 충분히 적용하고 있는지에 대한 공통 질문에서, 비표준 분야에서는 “그렇다”가 6.7%, “약간 그렇다”가 46.7%로 조사되었다. 세부적으로 “아키텍처 설계 시 사용하는 기법”을 선택하라는 질문에 비표준 분야는 “모듈방식”과 “구조 다이어그램”이라는 응답이 가장 많이 조사되었다.

<표 4-67> 비표준 분야 아키텍처 설계 시 사용하는 기법

활동	건수	%
오류 감지	4	27%
코드 감지 오류	4	27%
오류 주장 프로그래밍	1	7%
다양한 모니터 기술	5	33%
중복 및 가용성 설계	5	33%
모듈 방식	11	73%
소프트웨어 안전 요구 사항 추적	3	20%
구조 다이어그램	9	60%
컴퓨터 지원 사양 및 설계 도구	4	27%
이벤트 중심, 최대 응답 시간 보장	3	20%
정적 리소스 할당 / 동기화	4	27%

- 자동차 분야

아키텍처 설계 단계에서 안전성 보증을 위한 기법을 충분히 적용하고 있는지에 대한 공통 질문에서, 자동차 분야의 경우 “그렇다” 가 30%, “약간 그렇다” 가 20%로 조사되었다.

세부적으로 “아키텍처 설계 시 사용하는 기법” 을 선택하라는 질문에 자동차 분야는 “비정형 표기법(그림, 다이어그램, 단순 도표 등)” , “준정형 표기법(제어흐름도, 데이터 흐름도, 상태천이도, UML등)” 을 가장 많이 사용하는 것으로 조사되었다.

<표 4-68> 자동차 분야 아키텍처 설계 시 사용하는 기법

활동	건수	%
자연어	6	55%
비정형 표기법(그림, 다이어그램, 단순 도표 등)	8	73%
준정형 표기법(제어흐름도, 데이터 흐름도, 상태천이도, UML등)	7	64%
정형 표기법 (Zed, PVS, VMD등)	1	9%
시스템 안전 요구사항과 SW안전 요구사항 간의 추적	4	36%
안전 요구사항 관리 자동화 Tool 사용	2	18%

“아키텍처 설계 시 검증 방법” 에 대한 질문에서 자동차 분야는 “설계에 대한 워크스루

(walk-through)”, “설계에 대한 인스팩션(Inspection)” 으로 검증한다는 응답이 가장 많았다.

<표 4-69> 자동차 분야 아키텍처 설계 시 검증 방법

활동	건수	%
설계에 대한 워크스루(walk-through)	8	73%
설계에 대한 인스펙션(inspection)	7	64%
설계의 동적 부분에 대한 시뮬레이션	4	36%
프로토타입 제작	2	18%
정형 검증	0	0%
제어 흐름 분석, 데이터 흐름 분석	5	45%
자동화 도구 사용 검증	1	9%

- 철도 분야

아키텍처 설계 단계에서 안전성 보증을 위한 기법을 충분히 적용하고 있는지에 대한 공통 질문에서 철도분야는 “그렇다, 매우 그렇다” 가 63.7%, “약간 그렇다” 가 18.2%로 조사되었다.

세부적으로 “아키텍처 설계 시 사용하는 기법” 을 선택하라는 질문에 철도 분야는 “결함검출&진단” 이라는 응답이 가장 많았다.

<표 4-70> 철도 분야 아키텍처 설계 시 사용하는 기법

활동	건수	%
방어적 프로그래밍	6	55%
결함 검출 & 진단	9	82%
오류 정정 코드, 오류 검출 코드	6	55%
고장 단정 프로그래밍	2	18%
다양화 프로그래밍	3	27%
복구 블록	2	18%
역방향 복구, 전방향 복구	1	9%
고장 복구 재시도 방법	2	18%
소프트웨어의 동적 재구성	2	18%
소프트웨어 오류 영향 분석	4	36%
정보 은닉, 정보 캡슐화	4	36%
완전하게 정의된 인터페이스	5	45%
모델링	6	55%
구조적 방법론	8	73%
컴퓨터 지원 설계 및 명세도구를 통한 모델링	1	9%

추가적인 활동으로 “소프트웨어 설계를 위해 적용하는 대책 및 기법”에 대한 조사로 “구조적 방법론과 프로그래밍”이라는 응답이 가장 많았다.

<표 4-71> 철도 분야 소프트웨어 설계를 위해 적용하는 대책 및 기법

활동	건수	%
상태 전이 다이어그램	5	45%
시퀀스 다이어그램	8	73%
구조적 방법론	9	82%
분석 가능한 프로그램	2	18%
엄격한 형식의 프로그래밍 언어	4	36%
구조적 프로그래밍	9	82%
절차적 프로그래밍	3	27%

“소프트웨어 통합 테스트 명세를 위해 적용하는 대책 및 기법”에 대한 조사로 “기능·블랙 박스 테스트”와 “성능테스트”라는 응답이 가장 많았다.

<표 4-72> 소프트웨어 통합 테스트 명세를 위해 적용하는 대책 및 기법

활동	건수	%
동적 분석 및 테스트	8	73%
기능 / 블랙박스 테스트	9	82%
성능 테스트	9	82%

“소프트웨어 / 하드웨어 통합 테스트 명세를 위해 적용하는 대책 및 기법”에 대한 조사에서는 “인터페이스 시험”이라는 응답이 가장 많았다.

<표 4-73> 소프트웨어/하드웨어 통합 테스트 명세를 위해 적용하는 대책 및 기법

활동	건수	%
정적 분석	6	55%
동적 분석 및 시험	7	64%
추적성	7	64%
소프트웨어 오류 영향 분석	7	64%
코드 시험 적용범위	3	27%
기능적/블랙박스 시험	8	73%
성능 시험	8	73%
인터페이스 시험	10	91%

- 항공분야

아키텍처 설계 단계에서 안전성 보증을 위한 기법을 충분히 적용하고 있는지에 대한 공통 질문에서 항공 분야의 경우 “그렇다, 매우 그렇다”가 27.3%, “약간 그렇다”가 27.3%로 파악되었다. 세부적으로 “아키텍처 설계 시 사용하는 기법”을 선택하라는 질문에 항공 분야는 “분할 기법(Partitioning)”이라는 응답이 가장 많았다.

<표 4-74> 항공 분야 아키텍처 설계 시 사용하는 기법

활동	건수	%
분할 기법(Partitioning)	8	62%
다수 버전의 상이한 소프트웨어(Multiple-version Dissimilar Software)	2	15%
안전 모니터링(Safety Monitoring)	5	38%

산업 분야를 종합해보면 철도가 가장 수준이 높았으며 자동차, 항공 순으로 높은 수준을 보였다. 비표준 분야는 타 분야에 비해 수준이 상대적으로 낮았다. 평균적으로는 “그렇다, 매우 그렇다”가 31.9%, “약간 그렇다”가 28%로 나타났다.

비표준 분야는 요구사항이 아키텍처 설계에 어느 정보 반영되는지 현황을 묻는 공통 질문에 긍정적인 답을 했으나, 기법을 사용하느냐는 세부 질문 결과 기법 사용률이 부진하여 전 질문에 대한 해석을 조금 다르게 할 수 있다. 철도나 항공 등 다른 분야보다는 프로세스 적용이나 안전 기대 수준이 낮아 긍정적이 답이 나왔음을 유추할 수 있다. 아키텍처 설계는 아직은 비정형 표기법을 이용하며, 구조적 방법론을 가장 많이 사용하는 것으로 조사되었다.

6. 귀사는 아키텍처 명세 단계의 산출물이 상세 설계 및 구현에 필요한 정보를 충분히 제공하고 있습니까?

1. 전혀 아니다 2. 아니다 3. 약간 아니다 4. 약간 그렇다 5. 그렇다 6.매우 그렇다

<표 4-75> 아키텍처 명세서가 상세설계와 구현에 필요한 정보 제공 현황 분석

(단위 : 백분율)

산업분야	전혀 아니다		아니다		약간 아니다		약간 그렇다		그렇다		매우 그렇다	
비표준	1	6.7	1	6.7	2	13.3	8	53.3	3	20.0	0	0.0
자동차	1	9.1	1	9.1	3	27.3	2	18.2	4	36.4	0	0.0
철도	0	0.0	0	0.0	1	9.1	3	27.3	5	45.5	2	18.2
항공	-	0.0	-	0.0	1	7.7	4	30.8	7	53.8	1	7.7
평균	1	2.9	1	2.9	5	14.3	9	25.7	16	45.7	3	8.6

- 비표준 분야

아키텍처 명세 단계의 산출물이 상세 설계 및 구현에 필요한 정보를 충분히 제공하고 있는지에 대한 공통 질문에서, 비표준 분야에서는 “그렇다”가 20%, “약간 그렇다”가 53.3%로 조사되었다. 세부적으로 “아키텍처 설계 관련 작성하는 산출물 목록”을 선택하라는 질문에 비표준 분야는 “안전무결성을 유지할 설계 유형 선택(입출력/통신/인터페이스/유지보수 자료 등)”이라는 응답이 가장 많이 조사되었다.

<표 4-76> 비표준 분야 아키텍처 설계 관련 작성하는 산출물 목록

산출물	건수	%
소프트웨어 안전 요구사항 명세를 만족하는 통합적 기법과 수단 선택, 그 근거 입증	3	20
컴포넌트/서브시스템으로 분할하는 것에 대한 정보(신규/검증 여부, 안전무결성)	5	33
소프트웨어/하드웨어 상호작용 결정, 평가, 기술	7	47
명확한 아키텍처 표현법 사용.	6	40
안전무결성을 유지할 설계 유형 선택(입출력/통신/인터페이스/유지보수 자료 등)	8	53
안전 요구사항 명세를 충족하는 적절한 소프트 웨어 아키텍처 통합 시험 계획	4	27

- 자동차 분야

아키텍처 명세 단계의 산출물이 상세 설계 및 구현에 필요한 정보를 충분히 제공하고 있는지에 대한 공통 질문에서, 자동차 분야의 경우 “그렇다”가 36.4%, “약간 그렇다”가 18.2%로 조사되었다.

세부적으로 “아키텍처 설계 관련 작성하는 산출물 목록”을 선택하라는 질문에 자동차

분야는 “소프트웨어 아키텍처 설계 명세서” 작성이 가장 많은 것으로 조사되었다.

<표 4-77> 자동차 분야 아키텍처 설계 관련 작성하는 산출물 목록

산출물	건수	%
소프트웨어 아키텍처 설계 명세서	8	73
안전 계획 (갱신)	4	36
소프트웨어 안전 요구사항 명세서 (갱신)	5	45
안전 분석 보고서	5	45
종속고장 분석 보고서	2	18
소프트웨어 검증 보고서 (갱신)	7	64

- 철도 분야

아키텍처 명세 단계의 산출물이 상세 설계 및 구현에 필요한 정보를 충분히 제공하고 있는지에 대한 공통 질문에서 철도분야는 “그렇다, 매우 그렇다” 가 63.7%, “약간 그렇다” 가 27.3%로 조사되었다.

세부적으로 “아키텍처 설계 관련 작성하는 산출물 목록” 을 선택하라는 질문에 철도 분야는 “소프트웨어 설계 명세서” 와 “소프트웨어 통합테스트 명세서” 라는 응답이 가장 많았다. 기타 의견으로 고객 요구시 “소프트웨어 모듈 명세서” 를 작성하는 경우 있다고 조사되었다.

<표 4-78> 철도 분야 아키텍처 설계 관련 작성하는 산출물 목록

산출물	건수	%
소프트웨어 아키텍처 명세서	9	82
소프트웨어 설계 명세서	10	91
소프트웨어 인터페이스 명세서	9	82
소프트웨어 통합테스트 명세서	10	91
소프트웨어/하드웨어 통합테스트 명세서	9	82
소프트웨어 아키텍처 및 설계 검증 보고서	6	55

- 항공분야

아키텍처 명세 단계의 산출물이 상세 설계 및 구현에 필요한 정보를 충분히 제공하고 있는지에 대한 공통 질문에서 “그렇다, 매우 그렇다”가 63.6%로 파악되었다.

세부적으로 “아키텍처 설계 관련 산출물에 포함되는 항목”을 선택하라는 질문에 항공분야는 “소프트웨어 구성 요소에 대한 설명)”이라는 응답이 가장 많았다. 기타 의견으로 방위사업청에 제출하는 산출물로 작성한다는 의견도 조사 되었다.

<표 4-79> 항공 분야 아키텍처 설계 관련 산출물에 포함되는 항목

항목	건수	%
안전 관련 시스템 요구 사항을 추적 할 수있는 설계 결정에 대한 이론적 근거	9	69
소프트웨어 로딩, 사용자 수정이 가능한 소프트웨어 또는 여러 버전의 다른 소프트웨어와 같은 구현 방법에 대한 설계 방법 및 세부 정보	5	38
파티션 분할 방법 및 파티션 위반 방지 방법	2	15
소프트웨어 구성 요소에 대한 설명 (새로운 구성인지 또는 이전에 개발되었는지, 이전에 개발된 경우, 취해진 기준을 참조).	10	77

아키텍처 설계 관련 산출물은 아키텍처 설계 명세서가 가장 많이 작성하였으며, 철도분야에서는 소프트웨어 통합테스트 명세서도 중요시 여기는 것으로 조사되었다. 비표준 분야는 상대적으로 아키텍처 설계 관련 산출물 작성 비율이 낮았다.

3) 상세 설계 단계

7. 상세 설계과정에서 구현을 위한 충분한 설계가 이루어지고 있다고 생각하십니까?

1. 전혀 아니다 2. 아니다 3. 약간 아니다 4. 약간 그렇다 5. 그렇다 6.매우 그렇다

<표 4-80> 상세설계 과정이 구현을 위한 충분한 설계가 이루어지는지 현황 분석

(단위 : 건수, 백분율)

산업분야	전혀 아니다		아니다		약간 아니다		약간 그렇다		그렇다		매우 그렇다	
비표준	1	6.7	4	26.7	-	0.0	5	33.3	5	33.3	-	0.0
자동차	1	9.1	-	0.0	4	36.4	2	18.2	4	36.4	-	0.0
철도	-	0.0	-	0.0	1	9.1	2	18.2	7	63.6	1	9.1
항공	-	0.0	-	0.0	3	23.1	6	46.2	3	23.1	1	7.7
평균	1	2.9	-	0.0	8	22.9	10	28.6	14	40.0	2	5.7

- 비표준 분야

상세 설계과정에서 구현을 위한 충분한 설계가 이루어지고 있는지에 대한 공통 질문에서, 비표준 분야에서는 “그렇다” 가 33.3%, “약간 그렇다” 가 33.3%로 조사되었다.

세부적으로 “상세 설계를 위해 수행하는 활동” 을 선택하라는 질문에 비표준 분야는 “소프트웨어 안전을 위한 요구사항 명세, 소프트웨어 아키텍처 설계, 소프트웨어 안전을 확증하기 위한 계획 등의 정보를 상세 설계 시작 전 완료” 라는 응답이 가장 많이 조사되었다.

<표 4-81> 비표준 분야 상세 설계를 위해 수행하는 활동

활동	건수	%
소프트웨어 안전을 위한 요구사항 명세, 소프트웨어 아키텍처 설계, 소프트웨어 안전을 확증하기 위한 계획 등의 정보를 상세 설계 시작 전 완료	10	67%
소프트웨어는 모듈성, 시험 가능성, 안전한 변경을 위한 능력을 갖추도록 제작	7	47%
각 소프트웨어 모듈의 설계와 각 소프트웨어 모듈에 적용되어야 할 시험방법을 명시	7	47%

“상세 설계를 위해 수행하는 안전관련 활동” 에 대한 분석으로는 비표준 분야에서는 “소프트웨어와 시스템 간 인터페이스” 라는 응답이 가장 많았다.

<표 4-82> 비표준 분야 상세 설계를 위해 수행하는 안전관련 활동

활동	건수	%
시스템이 안전한 상태를 달성 유지할 수 있도록 하는 기능	6	40%
하드웨어에서 결함을 검출, 알림, 관리하는 기능	8	53%
센서, 액추에이터 결함을 검출, 알림, 관리하는 기능	8	53%

소프트웨어 스스로 결함을 검출, 알림, 관리하는 기능	2	13%
안전기능에 대한 온오프라인상의 정기적 테스트 관련 기능	0	0%
소프트웨어와 시스템 간 인터페이스	9	60%
용량과 응답시간 성능	3	20%

"안전 무결성 요구사항이 만족함을 보증할 단위/통합 시험 계획을 수립" 에 대한 질문에 대해 "약간 그렇다" 라는 응답이 6건으로 가장 많았다.

<표 4-83> 안전 무결성 요구사항이 만족함을 보증할 단위/통합 시험 계획을 수립

(단위 : 건수, 백분율)

산업분야	전혀 아니다		아니다		약간 아니다		약간 그렇다		그렇다		매우 그렇다	
비표준	2	13	3	20	1	7	6	40	3	20	0	0

- 자동차 분야

상세 설계과정에서 구현을 위한 충분한 설계가 이루어지고 있는지에 대한 공통 질문에서, 자동차 분야의 경우 "그렇다" 가 36.4%, "약간 그렇다" 가 18.2%로 조사되었다.

세부적으로 "유닛 설계를 위해 수행하는 활동" 을 선택하라는 질문에 자동차 분야는 "비 안전 관련 요구사항 설계" 와 "각 소프트웨어 유닛의 기능 (pseudo code, flow chart 등을 사용) 상세 설계" 작성이 가장 많은 것으로 조사되었다.

<표 4-84> 자동차 분야 유닛 설계를 위해 수행하는 활동

활동	건수	%
비 안전 관련 요구사항 설계	7	64%
각 소프트웨어 유닛의 기능 (pseudo code, flow chart 등을 사용) 상세 설계	7	64%
각 소프트웨어 유닛의 인터페이스 (function prototype 등) 설계	6	55%
각 소프트웨어 유닛의 execution trigger 조건 설계	3	27%
각 소프트웨어 유닛이 사용하는 data 및 data의 structure 설계	5	45%
안전관련 요구사항의 각 소프트웨어 유닛의 ASIL 설계	3	27%

"개발 언어는 무엇을 사용하고 있습니까?" 라는 질문에서 자동차 분야는 "C언어 (MISRA-C coding rule 적용) 개발" 이라는 응답이 가장 많았다. 기타 개발 언어로 C++, c#, JAVA 언어를 사용하는 것으로 조사되었다.

<표 4-85> 자동차 분야 사용 하는 개발 언어

사용 개발 언어	건수	%
C언어(MISRA-C coding rule 적용) 개발	10	91%
그래픽 모델링 방식 개발	1	9%
어샘플러 프로그래밍	1	9%
혼합 개발 언어 사용	0	0%

“유닛 설계 속성(가독성, 단순성, 강건성 등)을 달성하기 위한 설계 원칙을 준수 활동”에서는 “변수 이름을 다목적으로 사용하지 않음”이라는 응답이 가장 많았다. 기타 의견으로 “MISRA-C에 의한 제약 반영”을 수행하는 것으로 조사되었다.

<표 4-86> 유닛 설계 속성을 달성하기 위한 설계 원칙을 준수 활동

설계 원칙 준수 활동	건수	%
변수 초기화 설계	6	55%
서브프로그램과 함수에서 하나의 진입점과 하나의 종료점 설계	4	36%
동적 객체 또는 변수를 사용하지 않음. 사용한다면 동적 변수 생성시 온라인 시험 설계	3	27%
변수 이름을 다목적으로 사용하지 않음	8	73%
전역 변수를 사용하지 않음, 만약 사용해야 한다면 그 사용에 대해 정당성 명세	6	55%
포인터의 제한된 사용 설계	6	55%
묵시적 형 변환 없이 설계	4	36%
숨겨진 데이터 흐름이나 제어 흐름 없이 설계	1	9%
무조건적 점프 없이 설계	5	45%
숨겨진 데이터 흐름이나 제어 흐름 없도록 설계	2	18%
재귀호출이 없도록 설계	5	45%

- 철도 분야

상세 설계과정에서 구현을 위한 충분한 설계가 이루어지고 있는지에 대한 공통 질문에서 철도분야는 “그렇다, 매우 그렇다”가 72.7%, “약간 그렇다”가 18.2%로 조사되었다.

세부적으로 “소프트웨어 컴포넌트 설계 명세를 위해 수행하는 활동”을 선택하라는 질문에 철도 분야는 “소프트웨어 컴포넌트 설계 명세서를 작성”한다는 응답이 가장 많았다.

<표 4-87> 철도 분야 소프트웨어 컴포넌트 설계 명세를 위해 수행하는 활동

활동	건수	%
소프트웨어 컴포넌트 설계 명세서 작성	10	91%
소프트웨어 컴포넌트 설계 명세서는 가독성과 테스트 용이성을 고려하여 작성	7	64%
소프트웨어 컴포넌트 설계 시 크기와 복잡도에 대한 균형 고려	5	45%

철도 분야에서는 “소프트웨어 컴포넌트 설계 적합성 검토를 위해 수행하는 활동”에 대한 분석으로 “소프트웨어 컴포넌트 설계가 소프트웨어 설계 명세에서 도출되었는지 추적성을 검토한다.”는 응답이 가장 많았다.

<표 4-88> 철도 분야 아키텍처 설계를 위해 수행하는 안전관련 활동

활동	건수	%
소프트웨어 컴포넌트 설계 적합성은 품질보증계획에 따라 작성자와 별도 조직에서 검토	6	55%
소프트웨어 컴포넌트 설계가 소프트웨어 설계 명세에서 도출되었는지 추적성을 검토한다.	10	91%
소프트웨어 컴포넌트 설계 명세의 정확성/완전성, 일관성, 구현 가능성, 표준 준수성 검토	8	73%

추가적인 활동으로 “소프트웨어 컴포넌트 테스트 명세를 위해 수행하는 활동”에 대한 조사로 “소프트웨어 컴포넌트 테스트 명세를 작성”한다는 응답이 가장 많았다.

<표 4-89> 소프트웨어 컴포넌트 테스트 명세를 위해 수행하는 활동

활동	건수	%
소프트웨어 컴포넌트 테스트 명세 작성	9	82%
블랙박스 테스트를 통해 컴포넌트가 의도된 기능을 수행한다는 것을 증명	7	64%
화이트박스 테스트를 통해 의도된 기능 수행을 위한 컴포넌트 내부의 상호작용 검토	7	64%
컴포넌트의 모든 부분이 테스트가 된다는 것을 증명	6	55%

“소프트웨어 컴포넌트 설계 검증을 위해 수행하는 활동”에 대한 질문에서는 “소프트웨어 컴포넌트 설계 검증 보고서를 기술”한다는 활동이 가장 많은 것으로 조사되었다.

〈표 4-90〉 소프트웨어 컴포넌트 설계 검증을 위해 수행하는 활동

활동	건수	%
소프트웨어 컴포넌트 설계 검증 보고서 기술	9	82%
소프트웨어 컴포넌트 설계 명세서의 가독성, 추적성 검증	8	73%
소프트웨어 컴포넌트 테스트 명세서는 소프트웨어 컴포넌트 설계 명세서의 모든 테스트 케이스를 기술했는지 검증	7	64%
소프트웨어 컴포넌트 테스트 명세서의 가독성, 추적성 검증	6	55%
소프트웨어 컴포넌트 설계 명세서의 요구사항의 구현 가능성, 테스트 가능성, 유지보수성 검증	7	64%

- 항공분야

상세 설계과정에서 구현을 위한 충분한 설계가 이루어지고 있는지에 대한 공통 질문에서 항공 분야의 경우 “그렇다, 매우 그렇다”가 27.3%, “약간 그렇다”가 54.5%로 파악되었다. 세부적으로 “상세 설계를 위해 수행하는 활동”을 선택하라는 질문에 항공 분야는 “상위수준 요구사항을 모두 반영하여 상세 요구사항을 정의”한다는 응답이 가장 많았다.

〈표 4-91〉 항공 분야 상세 설계를 위해 수행하는 활동

활동	건수	%
상위 수준 요구 사항을 모두 반영하여 상세 요구사항을 정의	12	92%
상위 수준 요구 사항과 상세 요구 사항 간의 양방향 연관을 보여주는 추적성 확보	10	77%
소프트웨어 설계 표준을 준수하여 상세 요구 사항을 명세	6	46%

또한, 항공분야에서 “상세 설계를 위해 수행하는 안전관련 활동”에 대한 질문에서는 “실패 조건에 대한 대응이 안전 관련 요구사항과 일치하는지 확인”한다는 응답이 가장 조사되었다.

<표 4-92> 항공 분야 상세 설계를 위해 수행하는 안전관련 활동

활동	건수	%
소프트웨어 설계에서 파티션 또는 기타 아키텍처 수단을 사용하여, 소프트웨어의 일부 구성 요소에 대한 소프트웨어 레벨 할당이 변경된 경우, 추가적인 데이터는 파생된 요구 사항으로 정의	4	31%
파생된 상세 요구 사항은 시스템 안전성 평가 프로세스와 시스템 프로세스에 제공	3	23%
수정 불가능한 구성 요소의 안전한 작동에 방해가 되지 않도록 수정 가능 구성 요소로부터 보호 될 수 있는 요구사항을 정의	1	8%
안전 관련 요구사항은 제어 흐름 및 데이터 흐름을 주시하여 상세 요구사항을 정의함	5	38%
실패 조건에 대한 대응이 안전 관련 요구 사항과 일치하는지 확인	6	46%

"안전관련 상세 요구사항 검증과 관련된 활동" 에 대한 조사로 "상세 요구사항이 상위수준 요구사항을 충족시키고, 설계 기준을 올바르게 정의하였는지 검증" 한다는 응답이 가장 많았다.

<표 4-93> 항공 분야 안전관련 상세 요구사항 검증과 관련된 활동

활동	건수	%
상세 요구 사항이 상위 수준 요구 사항을 충족시키고, 설계 기준을 올바르게 정의하였는지 검증	12	92%
상세 요구사항이 정확하고 모호하지 않으며 서로 충돌되지 않음을 검증	6	46%
상세 요구 사항과 대상 컴퓨터의 하드웨어 /소프트웨어간에 충돌이 없는지 검증 (특히 버스 로드의 자원 사용, 시스템 응답시간, 하드웨어 입/출력)	6	46%
상위 수준 요구 사항과 상세 요구 사항과 추적이 가능한지 검증	11	85%

비표준 분야는 상세 질문 분석 결과, 아키텍처 설계 단계보다는 상세 설계 분야의 활동은 좀 더 많은 기업이 하고 있는 것으로 조사되었다. 상세 설계 단계에서도 상위 요구사항을 반영하여 상세 요구사항을 정의하는 것이 중요하며, 이에 따라 추적성 검토 활동은 필수적으로 해야 하는 활동으로 조사되었다.

산업 분야별 "SW 구현을 위한 충분한 설계가 이루어지고 있지 않는 이유" 에 대해 조사 결과 분야별로 다음과 같은 의견들이 조사되었다.

<표 4-94> SW 구현을 위한 충분한 설계가 이루어지고 있지 않는 이유 분석

분야	응답
비표준	• 인력 부족 / 시간 비용 부족 • SW공학 모름 • 업무일정, 관련자들의 공감대 부족
자동차	• 일정, 인력 부족 (분석단계에서 시간이 꽤 소요됨) • 코드의 복잡성 • 설계 개념 이해 부족
철도	• 일정촉박, 개발인력의 개발일정 부족, 인력과 시간부족 • 전문적 지식 부족, 인식결여
항공	• 프로젝트 구성원간의 이해 차이(Domain에 대한 상이한 이해) • 인력, 비용, 일정 문제와 함께 소스 구현으로 바로 진입하는 개발 관행 문제 • SW개발 프로세스 각 단계별 세부적인 프로세스 진행 지침서가 부족함 • 고객요구조건의 모호성, 개발인력 부족

8. 상세 설계 단계에서 안전성 보증을 위한 기법을 충분히 적용하고 있다고 생각하십니까?

1. 전혀 아니다 2. 아니다 3. 약간 아니다 4. 약간 그렇다 5. 그렇다 6.매우 그렇다

<표 4-95> 상세설계 단계에서 안전성 보증 기법을 어느 정도 적용되는지 현황 분석

(단위 : 백분율)

산업분야	전혀 아니다		아니다		약간 아니다		약간 그렇다		그렇다		매우 그렇다	
비표준	2	14.3	2	14.3	3	21.4	5	35.7	2	14.3	-	0.0
자동차	1	9.1	2	18.2	2	18.2	3	27.3	3	27.3	-	0.0
철도	-	0.0	-	0.0	3	27.3	2	18.2	5	45.5	1	9.1
항공	-	0.0	2	15.4	4	30.8	4	30.8	2	15.4	1	7.7
표준합계	1	2.9	4	11.4	9	25.7	9	25.7	10	28.6	2	5.7

- 비표준 분야

상세 설계 단계에서 안전성 보증을 위한 기법을 충분히 적용하고 있는지에 대한 공통 질문에서, 비표준 분야에서는 “그렇다” 가 14.3%, “약간 그렇다” 가 35.7%로 조사되었다.

세부적으로 “개발사용 도구의 안전성 검증을 위해 검토하는 내용” 을 선택하라는 질문에 “검증된 프로그래밍 언어, 개발 지원 및 테스트 지원 도구 사용” 이라는 응답이 가장 많았다.

<표 4-96> 비표준 분야 개발사용 도구의 안전성 검증을 위해 검토하는 내용

검토 내용	건수	%
검증된 프로그래밍 언어, 개발 지원 및 테스트 지원 도구 사용	10	67%
사용하는 프로그래밍 언어가 인증된 제품인지 검토	7	47%
사용하는 프로그래밍 언어가 안전 무결성 수준을 만족하는지 검토	2	13%
증빙을 위한 소프트웨어 소스 코드 문서화 정책, 프로그래밍 언어 특성 파악	7	47%
프로그래밍 언어의 널리 알려진 단점 해소방안 마련 여부 검토	2	13%

“안전 무결성 수준에 따른 설계를 위해 사용하는 기법” 에 대한 분석으로는 비표준 분야에서는 “모듈 방식의 소프트웨어 개발” 이라는 응답이 가장 많았다.

<표 4-97> 비표준 분야 안전 무결성 수준에 따른 설계를 위해 사용하는 기법

기법	건수	%
컴퓨터 지원 설계 도구의 사용	5	33%
무결성 수준 만족을 위한 방어적 프로그래밍 추가	2	13%
모듈 방식의 소프트웨어 개발	12	80%
디자인 및 코딩 표준 마련 및 준수 모니터링	5	33%

“정형화된 모델링을 위해 사용하는 기법” 에 대한 분석으로는 비표준 분야에서는 “논리・기능블록다이어그램” 과 “데이터 흐름도” 라는 응답이 가장 많았다.

<표 4-98> 비표준 분야 정형화된 모델링을 위해 사용하는 기법

기법	건수	%
논리 / 기능 블록 다이어그램	9	60%

시퀀스 다이어그램	5	33%
상태 전이 다이어그램	5	33%
데이터 흐름도	9	60%
ERD	6	40%
UML	6	40%

- 자동차 분야

상세 설계 단계에서 안전성 보증을 위한 기법을 충분히 적용하고 있는지에 대한 공통 질문에서, 자동차 분야의 경우 “그렇다”가 27.3%, “약간 그렇다”가 27.3%로 조사되었다. 세부적으로 “SW유닛 설계 검증 기법”을 선택하라는 질문에 자동차 분야는 “설계에 대한 워크스루(walk-through)”와 “설계에 대한 인스팩션(inspection)”이 가장 많은 것으로 조사되었다.

<표 4-99> 자동차 분야 SW유닛 설계 검증 기법

기법	건수	%
설계에 대한 워크스루(walk-through)	8	73%
설계에 대한 인스펙션(inspection)	8	73%
준정형 검증	3	27%
정형 검증	0	0%
제어 흐름 분석	5	45%
데이터 흐름 분석	4	36%
정적 코드 분석	7	64%
의미적 코드 분석	1	9%
자동화 도구 사용 검증	4	36%

- 철도 분야

상세 설계 단계에서 안전성 보증을 위한 기법을 충분히 적용하고 있는지에 대한 공통 질문에서 철도분야는 “그렇다, 매우 그렇다”가 54.6%, “약간 그렇다”가 18.2%로 조사되었다. 세부적으로 “소프트웨어 컴포넌트 설계 기법”을 선택하라는 질문에 철도 분야는 “설계 및 코딩 표준”이라는 응답이 가장 많았다.

<표 4-100> 철도 분야 소프트웨어 컴포넌트 설계 기법

기법	건수	%
정형 명세 (Formal Specification)	4	36%
데이터 흐름 다이어그램	8	73%
모듈 방식	8	73%
컴포넌트	6	55%
정보 은닉화	1	9%
정보 캡슐화	3	27%
매개 변수 개수 제한	4	36%
설계 및 코딩 표준	11	100%
분석 가능한 프로그램	3	27%
엄격한 형식의 프로그래밍 언어	4	36%
구조적 프로그래밍	7	64%
프로그래밍 언어	4	36%
언어 하위 집합	1	9%
객체 지향 언어	2	18%
절차적 프로그래밍	4	36%
메타 프로그래밍	3	27%

- 항공분야

상세 설계 단계에서 안전성 보증을 위한 기법을 충분히 적용하고 있는지에 대한 공통 질문에서 항공 분야의 경우 “그렇다, 매우 그렇다”가 27.3%, “약간 그렇다”가 27.3%로 파악되었다.

세부적으로 “상세 설계 검증 기법”을 선택하라는 질문에 항공 분야는 “제어 흐름 분석”이라는 응답이 가장 많았다. 검증 기법에 사용하고 있는 자동화 도구로는 LDRA, QAC, Polyspace, Codesonar, Vector CAST를 사용하는 것으로 조사되었다.

<표 4-101> 상세 설계 검증 기법

기법	건수	%
설계에 대한 워크스루(walk-through)	6	46%

설계에 대한 인스펙션(inspection)	8	62%
준정형 검증	2	15%
정형 검증	1	8%
제어 흐름 분석	10	77%
데이터 흐름 분석	9	69%
정적 코드 분석	8	62%
의미적 코드 분석	1	8%
자동화 도구 사용 검증	5	38%

상세 설계 검증을 위해서는 정형 검증이나 준정형 검증 보다는 설계에 의한 워크스루 방식을 많이 사용했다. 설계 방식은 모듈 방식을 많이 사용하였다. 그러나 항공 분야에서는 제어 흐름 분석, 데이터 흐름 분석 등 상세 설계 자체를 분석하는 경향도 보였다.

9. 상세 설계 단계의 산출물이 구현에 필요한 정보를 충분히 제공하고 있습니까?

1. 전혀 아니다 2. 아니다 3. 약간 아니다 4. 약간 그렇다 5. 그렇다 6.매우 그렇다

<표 4-102> 상세설계 단계 산출물이 구현에 필요한 정보를 제공하는지 현황 분석

(단위 :건수, 백분율)

산업분야	전혀 아니다		아니다		약간 아니다		약간 그렇다		그렇다		매우 그렇다	
비표준	1	6.7	2	13.3	1	6.7	5	33.3	6	40.0	-	0.0
자동차	1	9.1	-	0.0	4	36.4	2	18.2	4	36.4	-	0.0
철도	-	0.0	-	0.0	1	9.1	3	27.3	5	45.5	2	18.2
항공	-	0.0	-	0.0	1	7.7	7	53.8	4	30.8	1	7.7
합계	1	2.9	-	0.0	6	17.1	12	34.3	13	37.1	3	8.6

- 비표준 분야

상세 설계 단계의 산출물이 구현에 필요한 정보를 충분히 제공하고 있는지에 대한 공통 질문에서, 비표준 분야에서는 “그렇다”가 40%, “약간 그렇다”가 33.3%로 조사되었다. 세부적으로 “상세 설계 관련 작성하는 산출물 목록”을 선택하라는 질문에 비표준 분야는

“소프트웨어 인터페이스 명세서” 라는 응답이 가장 많이 조사되었다.

<표 4-103> 비표준 분야 상세 설계 관련 작성하는 산출물 목록

산출물	건수	%
소프트웨어 인터페이스 명세서	12	80%
소프트웨어 아키텍처 명세서	10	67%
명세서(소프트웨어 아키텍처 통합 시험, 계획)	6	40%
명세서(PE 하드웨어와 소프트웨어 통합 시험, 계획)	4	27%
명세서(소프트웨어 시스템 통합 시험, 계획)	7	47%
명세서(소프트웨어 모듈 시험, 계획)	7	47%
지침서(개발 도구와 코딩 매뉴얼)	6	40%

- 자동차 분야

상세 설계 단계의 산출물이 구현에 필요한 정보를 충분히 제공하고 있는지에 대한 공통 질문에서, 자동차 분야의 경우 “그렇다” 가 36.4%, “약간 그렇다” 가 18.2%로 조사되었다. 세부적으로 “유닛 설계 관련 작성하는 산출물 목록” 을 선택하라는 질문에 자동차 분야는 “소프트웨어 유닛(단위) 설계 명세서” 와 “소프트웨어 검증 명세서” 작성이 가장 많은 것으로 조사되었다.

<표 4-104> 자동차 분야 유닛 설계 관련 작성하는 산출물 목록

산출물	건수	%
소프트웨어 유닛(단위) 설계 명세서	7	64%
소프트웨어 검증 명세서	7	64%
소프트웨어 검증 보고서 (갱신)	5	45%

- 철도 분야

상세 설계 단계의 산출물이 구현에 필요한 정보를 충분히 제공하고 있는지에 대한 공통 질문에서 철도분야는 “그렇다, 매우 그렇다” 가 63.7%, “약간 그렇다” 가 27.3%로 조사되었다.

세부적으로 “컴포넌트 설계 단계에서 작성하는 산출물”을 선택하라는 질문에 철도 분야는 “소프트웨어 컴포넌트 설계 명세서”와 “소프트웨어 컴포넌트 테스트 명세서”라는 응답이 가장 많았다.

<표 4-105> 철도 분야 컴포넌트 설계 단계에서 작성하는 산출물

산출물	건수	%
소프트웨어 컴포넌트 설계 명세서	10	91%
소프트웨어 컴포넌트 테스트 명세서	10	91%
소프트웨어 컴포넌트 설계 검증 보고서	7	64%

- 항공분야

상세 설계 단계의 산출물이 구현에 필요한 정보를 충분히 제공하고 있는지에 대한 공통 질문에서 항공 분야의 경우 “그렇다, 매우 그렇다”가 27.3%, “약간 그렇다”가 63.6%로 파악되었다. 세부적으로 “상세 설계 관련 작성하는 산출물에 포함되는 항목”을 선택하라는 질문에 항공 분야는 “소프트웨어 아키텍처 전체에서 입·출력 설명”과 “설계의 데이터 흐름 및 제어 흐름”이라는 응답이 가장 많았다.

<표 4-106> 항공 분야 상세 설계 관련하여 작성하는 산출물에 포함되는 항목

활동	건수	%
상위 수준 요구 사항을 어떻게 충족시키는 지에 대한 자세한 설명	8	62%
소프트웨어 구조를 정의하는 소프트웨어 아키텍처에 대한 설명	11	85%
소프트웨어 아키텍처 전체에서 입/출력설명	12	92%
설계의 데이터 흐름 및 제어 흐름.	12	92%
리소스 제한, 각 리소스 및 제한 사항을 관리하는 전략, 마진 및 마진을 측정하는 방법	4	31%
일정한 절차 및 프로세서간/작업간 통신 메커니즘	8	62%
소프트웨어 설계 프로세스에서 비롯된 파생 요구 사항	4	31%
시스템에 비활성화된 코드가 포함되어 있는 경우 대상 컴퓨터에서 코드를 활성화 할 수 없도록 하는 방법에 대한 설명	2	15%

4) 구현 단계

10. 구현 과정에서 충분히 기능 구현 및 단위 시험이 이루어지고 있다고 생각하십니까?

1. 전혀 아니다 2. 아니다 3. 약간 아니다 4. 약간 그렇다 5. 그렇다 6.매우 그렇다

<표 4-107> SW 구현 시 기능 구현 및 단위 시험이 어느 정도 이루어지는지 현황 분석

(단위 : 건수, 백분율)

산업분야	전혀 아니다		아니다		약간 아니다		약간 그렇다		그렇다		매우 그렇다	
비표준	1	6.7	1	6.7	2	13.3	4	26.7	7	46.7	-	0.0
자동차	-	0.0	-	0.0	3	27.3	2	18.2	5	45.5	1	9.1
철도	-	0.0	1	9.1	1	9.1	-	0.0	8	72.7	1	9.1
항공	-	0.0	-	0.0	1	7.7	5	38.5	7	53.8	-	0.0
표준합계	-	0.0	1	2.9	5	14.3	7	20.0	20	57.1	2	5.7

- 비표준 분야

구현 과정에서 충분히 기능 구현 및 단위 시험이 이루어지고 있는지에 대한 공통 질문에서, 비표준 분야에서는 “그렇다”가 46.7%, “약간 그렇다”가 26.7%로 조사되었다.

세부적으로 “구현을 위해 수행하는 활동”을 선택하라는 질문에 비표준 분야는 “소스코드는 상위 문서를 기반으로 개발자 (Implementer)의 책임 하에 구현”이라는 응답이 가장 많이 조사되었다.

<표 4-108> 비표준 분야 구현을 위해 수행하는 활동

활동	건수	%
소스코드는 상위 문서를 기반으로 개발자 (Implementer)의 책임 하에 구현	10	67%
소스코드의 크기(size)와 복잡도(complexity)는 균형을 이루어 구현	3	20%
소스코드는 읽기 쉽고, 이해하기 쉬우며, 테스트 가능	8	53%
소스코드는 테스트가 시작되기 전에 형상관리 시스템에 등록하여 변경 이력 관리	4	27%

- 자동차 분야

구현 과정에서 충분히 기능 구현 및 단위 시험이 이루어지고 있는지에 대한 공통 질문에서, 자동차 분야의 경우 “그렇다” 가 54.6%, “약간 그렇다” 가 18.2%로 조사되었다.

세부적으로 “구현을 위해 수행하는 활동” 을 선택하라는 질문에 자동차 분야는 “소스코드는 상위 문서를 기반으로 개발자 (Implementer)의 책임 하에 구현” 과 “소스코드는 읽기 쉽고, 이해하기 쉬우며, 테스트 가능하게 구현” 이 가장 많은 것으로 조사되었다.

<표 4-109> 자동차 분야 구현을 위해 수행하는 활동

활동	건수	%
소스코드는 상위 문서를 기반으로 개발자 (Implementer)의 책임 하에 구현	7	64%
소스코드의 크기(size)와 복잡도(complexity)는 균형을 이루어 구현	4	36%
소스코드는 읽기 쉽고, 이해하기 쉬우며, 테스트 가능하게 구현	7	64%
소스코드는 테스트가 시작되기 전에 형상관리 시스템에 등록하여 변경 이력 관리	6	55%

- 철도 분야

구현 과정에서 충분히 기능 구현 및 단위 시험이 이루어지고 있는지에 대한 공통 질문에서 철도 분야는 “그렇다, 매우 그렇다” 가 81.8%로 높게 조사되었다.

세부적으로 “소프트웨어 컴포넌트 구현을 위해 수행하는 활동” 을 선택하라는 질문에 철도 분야는 “소스코드는 상위 문서를 기반으로 개발자 (Implementer)의 책임 하에 구현” 한다는 응답이 가장 많았다.

<표 4-110> 철도 분야 소프트웨어 컴포넌트 구현을 위해 수행하는 활동

활동	건수	%
소스코드는 상위 문서를 기반으로 개발자 (Implementer)의 책임 하에 구현	10	91%
소스코드의 크기(size)와 복잡도(complexity)는 균형을 이루어 구현	8	73%
소스코드는 읽기 쉽고, 이해하기 쉬우며, 테스트 가능	7	64%
소스코드는 테스트가 시작되기 전에 형상관리 시스템에 등록하여 변경 이력 관리	8	73%

“소프트웨어 컴포넌트 테스트를 위해 수행하는 활동” 에 조사에서는 “테스트 결과가 상위 설계서에 기술된 요구사항을 만족하는지 확인” 한다는 응답이 가장 많았다. 기타 의견으로 “정적 분석” 을 컴포넌트 테스트를 위해 수행하는 하는 것으로 조사되었다.

<표 4-111> 소프트웨어 컴포넌트 테스트를 위해 수행하는 활동

활동	건수	%
컴포넌트 테스트 수행 환경은 실제 동작환경 또는 최대한 근접한 환경에서 수행	6	55%
소프트웨어 컴포넌트 테스트 보고서는 테스터의 책임 하에 작성	6	55%
테스트 결과가 상위 설계서에 기술된 요구사항을 만족하는지 확인	11	100%
각 컴포넌트별로 테스트 커버리지 지표가 제공되고 요구 커버리지 수준 대비 달성율 기술	6	55%
할당된 SIL에 해당하는 테스트 커버리지 수준 만족	5	45%

추가적인 활동으로 “소프트웨어 컴포넌트 테스트 결과 평가를 위해 수행하는 활동” 에 대한 조사로 “소스코드 구현과 소프트웨어 컴포넌트 테스트 보고서 작성 후 소스코드 검증 보고서 기술” 한다는 응답이 가장 많았다.

<표 4-112> 소프트웨어 컴포넌트 테스트 결과 평가를 위해 수행하는 활동

활동	건수	%
구현의 적절성, 가독성, 추적성 등 검증	9	82%
소스코드 구현과 소프트웨어 컴포넌트 테스트 보고서 작성 후 소스코드 검증 보고서 기술	10	91%
소프트웨어 검증 보고서는 검증자(Verifier)의 책임 하에 작성	5	45%

- 항공분야

구현 과정에서 충분히 기능 구현 및 단위 시험이 이루어지고 있는지에 대한 공통 질문에서 항공 분야의 경우 “그렇다” 가 45.5%, “약간 그렇다” 가 45.5%로 파악되었다. 세부적으로 “각 SW 소스코드에서 검토할 항목” 을 선택하라는 질문에 항공 분야는 “소스코드가 소프트웨어 아키텍처에 정의된 데이터 흐름 및 제어 흐름과 일치하는지 검토” 한다는

응답이 가장 많았다.

<표 4-113> 항공 분야 소프트웨어 소스코드에서 검토할 항목

검토 항목	건수	%
소스 코드가 소프트웨어 아키텍처에 정의 된 데이터 흐름 및 제어 흐름과 일치하는지 검토	12	92%
소스 코드가 검증 할 수 없는 구문과 구조를 포함하지 않는지 검토	7	54%
코드 개발 중 소프트웨어 코드 표준 (예 : 복잡성 제한 및 코드 제약)을 준수하는지 검토	9	69%
상세 요구 사항이 소스 코드로 개발되었는지 검토	11	85%
소스 코드가 정확하고 완전하게 구현하였으며, 문서화되지 않은 기능을 구현하지 않았는지 검토	10	77%
자동 코드 생성기의 사용시 소프트웨어 계획 프로세스에서 정의 된 제약 조건을 따라 사용하는지 검토	2	15%

“소스코드는 상위 문서를 기반으로 개발자 (Implementer)의 책임 하에 구현” 이 구현을 위해 가장 많이 수행하는 활동이었다. 안전을 위해 중요한 활동인 “소스코드의 크기(size)와 복잡도(complexity)는 균형을 이루어 구현” 의 활동 비율은 낮았다.

SW 구현 과정에서 기능 구현 및 단위 시험이 충분히 이루어지고 있지 않는 이유” 에 대한 조사 결과 분야별로 다음과 같은 의견이 조사되었다.

<표 4-114> SW 구현 과정에서 기능 구현 및 단위 시험이 불충분한 이유 분석

분야	응답
비표준	• 인력 부족, 시간 부족 • 주요기능은 기 개발된 코드가 많아 단위시험이 생략됨 • SW공학 모름
자동차	• 일정, 인력 부족, 시간 및 비용 부족 • 개발자가 테스트하기 때문
철도	• 인력, 비용, 일정, 이해부족, 전문지식 부족 • 개발일정 준수로 인하여 충분히 시험되지 않음
항공	• 인력, 예산부족, 가용인력(Test 인력 = 예산 투입) • 단위시험까지는 너무 많은 비용 기간 소요됨, 기능 시험위주로만 진행 • 기능구현 및 단위 시험에 대한 세부적인 check-list등 정리된 절차 없음

11. 구현 단계에서 안전성 보증을 위한 기법을 충분히 적용하고 있다고 생각하십니까?

1. 전혀 아니다 2. 아니다 3. 약간 아니다 4. 약간 그렇다 5. 그렇다 6.매우 그렇다

<표 4-115> 구현 단계에서 안전성 보증을 기법을 적용 정도 현황 분석

(단위 : 건수, 백분율)

산업분야	전혀 아니다		아니다		약간 아니다		약간 그렇다		그렇다		매우 그렇다	
비표준	2	14.3	4	28.6	1	7.1	5	35.7	2	14.3	-	0.0
자동차	1	9.1	1	9.1	2	18.2	3	27.3	4	36.4	-	0.0
철도	-	0.0	1	9.1	1	9.1	3	27.3	5	45.5	1	9.1
항공	-	0.0	3	23.1	1	7.7	5	38.5	4	30.8	-	0.0
표준합계	1	2.9	5	14.3	4	11.4	11	31.4	13	37.1	1	2.9

- 비표준 분야

구현 단계에서 안전성 보증을 위한 기법을 충분히 적용하고 있는지에 대한 공통 질문에서, 비표준 분야에서는 “그렇다”가 14.3%, “약간 그렇다”가 35.7%로 조사되었다.

세부적으로 “각 소프트웨어 모듈의 안전성 확보를 위해 소스코드에서 검토해야 할 것들”을 선택하라는 질문에 비표준 분야는 “동적 객체, 변수 관련 코딩 표준”이라는 응답이 가장 많이 조사되었다.

<표 4-116> 각 소프트웨어 모듈의 안전성 확보를 위해 소스코드에서 검토 항목

검토 항목	건수	%
오류 가능성을 줄이기 위한 코딩 표준의 사용	7	47%
동적 객체 / 변수 관련 코딩 표준	10	67%
제한된 인터럽트 사용 관련 코딩 표준	6	40%
포인터의 사용 제한 관련 코딩 표준	5	33%
제한된 재귀 사용 관련 코딩 표준	5	33%
고수준 언어의 프로그램에서 구조화되지 않은 제어 흐름 제거 관련표준	4	27%

- 자동차 분야

구현 단계에서 안전성 보증을 위한 기법을 충분히 적용하고 있는지에 대한 공통 질문에서, 자동차 분야의 경우 “그렇다”가 36.4%, “약간 그렇다”가 27.3%로 조사되었다.

세부적으로 “소프트웨어 유닛 구현에 대한 적합한 시험 환경에 대한 구성”을 선택하라는 질문에 자동차 분야는 “SIL(software in the loop) 시험”이 가장 많은 것으로 조사되었다.

<표 4-117> 자동차 분야 소프트웨어 유닛 구현에 대한 적합한 시험 환경 구성

시험 환경	건수	%
MIL(model in the loop) 시험	4	36%
SIL(software in the loop)시험	6	55%
PIL(Processor in the loop) 시험	3	27%
HIL(Hardware in the loop) 시험	5	45%

“소프트웨어 유닛 구현 시 시험 방법에 대해 적용하고 있는 방법”에 대한 질문에서 자동차 분야는 “요구사항 기반 시험”이라는 응답이 가장 많았다.

<표 4-118> 자동차 분야 소프트웨어 유닛 구현 시 시험 방법에 대해 적용 방법

적용 방법	건수	%
요구사항기반시험	11	100%
인터페이스 시험	7	64%
결함 주입 시험 (예.변수의 변수 값 손상, 변이 코드 또는 CPU 레지스터의 값 손상등 주입)	5	45%
자원 사용 시험 (단, 에뮬레이터가 자원 사용 시험을 지원하는 경우에만 사용)	3	27%
모델과 코드 간 비교 시험(back-to-back test) (시뮬레이션할 수 있는 모델이 필요)	4	36%
측정을 위한 사용되는 코드나 디버깅 활용	4	36%

“소프트웨어 유닛 시험을 위한 시험 케이스 도출 방법 중 사용 중인 방법”에 대한 질문에서 자동차 분야는 “요구사항 분석”이라는 응답이 가장 많았다.

<표 4-119> 자동차 분야 소프트웨어 유닛 시험을 위한 시험 케이스 도출 방법

도출 방법	건수	%
요구 사항 분석	10	91%
등가 등급 생성 및 분석	6	55%
경계 값 분석	9	82%
오류 추측(전문가 판단에 의한 수집된 데이터 기준)	7	64%

- 철도 분야

구현 단계에서 안전성 보증을 위한 기법을 충분히 적용하고 있는지에 대한 공통 질문에서 철도분야는 “그렇다, 매우 그렇다”가 54.6%, “약간 그렇다”가 27.3%로 조사되었다.

세부적으로 “소프트웨어 컴포넌트 구현단계에 적용하는 기법”을 선택하라는 질문에 철도 분야는 “설계 및 코딩표준”이라는 응답이 가장 많았다.

<표 4-120> 철도 분야 소프트웨어 컴포넌트 구현단계에 적용하는 기법

기법	건수	%
정형 기법	2	18%
모델링	4	36%
구조적 방법론	7	64%
모듈 방식	7	64%
컴포넌트	4	36%
설계 및 코딩 표준	10	91%
분석 가능한 프로그램	4	36%
엄격한 형식의 프로그래밍 언어	4	36%
구조적 프로그래밍	6	55%
프로그래밍언어	4	36%
언어 하위집합 부록	1	9%
객체지향 프로그래밍	3	27%
절차적 프로그래밍	4	36%
메타프로그래밍	2	18%

- 항공분야

구현 단계에서 안전성 보증을 위한 기법을 충분히 적용하고 있는지에 대한 공통 질문에서 항공 분야의 경우 “그렇다”가 27.3%, “약간 그렇다”가 36.4%로 파악되었다.

세부적으로 “구현단계에 적용하는 기법”을 선택하라는 질문에 항공 분야는 “설계 및 코딩표준”이라는 응답이 가장 많았다.

<표 4-121> 항공 분야 구현단계에 적용하는 기법

기법	건수	%
모듈 방식	10	77%
컴포넌트	7	54%
설계 및 코딩 표준	11	85%
분석 가능한 프로그램	4	31%
엄격한 형식의 프로그래밍 언어	2	15%
구조적 프로그래밍	7	54%
프로그래밍 언어	6	46%
언어 하위집합 부록	0	0%
객체지향 프로그래밍	5	38%
절차적 프로그래밍	4	31%
메타프로그래밍	1	8%

“SW (컴포넌트) 구현단계에 안전성 보증 기법이 충분히 적용되지 않고 있는 이유”에 대한 조사 결과 분야별로 다음과 같은 의견이 조사되었다.

<표 4-122> 구현단계에 안전성 보증 기법이 충분히 적용되지 않고 있는 이유 분석

분야	응답
비표준	• 교육 및 인식 부족, 시간, 비용 부족, 일정 부족 • 대상이 없음, SW공학 모름, 안전성의 필요성이 아직은 낮음
철도	• 방어적 프로그래밍 기법 부족 • 기능구현이 우선적으로 적용됨 • 인력, 시간, 부족, 예산(제3자 검증)

12. 구현 단계의 산출물이 테스트에 필요한 정보를 충분히 제공하고 있습니까?

1. 전혀 아니다 2. 아니다 3. 약간 아니다 4. 약간 그렇다 5. 그렇다 6.매우 그렇다

<표 4-123> 구현 단계 산출물이 테스트에 필요한 정보 제공 정도에 대한 현황 분석

(단위 : 건수, 백분율)

산업분야	전혀 아니다		아니다		약간 아니다		약간 그렇다		그렇다		매우 그렇다	
비표준	2	13.3	2	13.3	1	6.7	5	33.3	5	33.3	-	0.0
자동차	1	9.1	-	0.0	2	18.2	4	36.4	4	36.4	-	0.0
철도	-	0.0	-	0.0	1	9.1	3	27.3	6	54.5	1	9.1
항공	-	0.0	2	15.4	-	0.0	5	38.5	5	38.5	1	7.7
표준합계	1	2.9	2	5.7	3	8.6	12	34.3	15	42.9	2	5.7

- 비표준 분야

구현 단계의 산출물이 테스트에 필요한 정보를 충분히 제공하고 있는지에 대한 공통 질문에서, 비표준 분야에서는 “그렇다”가 33.3%, “약간 그렇다”가 33.3%로 조사되었다.

- 자동차 분야

구현 단계의 산출물이 테스트에 필요한 정보를 충분히 제공하고 있는지에 대한 공통 질문에서, 자동차 분야의 경우 “그렇다”가 36.4%, “약간 그렇다”가 36.4%로 조사되었다.

세부적으로 “구현 단계에서 작성하는 산출물”을 선택하라는 질문에 자동차 분야는 “소프트웨어 유닛 테스트 보고서” 작성이 가장 많은 것으로 조사되었다.

<표 4-124> 구현 단계에서 작성하는 산출물

산출물	건수	%
소프트웨어 소스코드 및 지원 문서	8	73%
소프트웨어 유닛 테스트 보고서	9	82%
소프트웨어 소스코드 검증 보고서	7	64%

- 철도 분야

구현 단계의 산출물이 테스트에 필요한 정보를 충분히 제공하고 있는지에 대한 공통 질문에서 철도분야는 “그렇다, 매우 그렇다” 가 63.6%, “약간 그렇다” 가 27.3%로 조사되었다. 세부적으로 “컴포넌트 구현 단계에서 작성하는 산출물” 을 선택하라는 질문에 철도 분야는 “소프트웨어 소스코드 및 지원문서” 라는 응답이 가장 많았다.

<표 4-125> 컴포넌트 구현 단계에서 작성하는 산출물

산출물	건수	%
소프트웨어 소스코드 및 지원 문서	9	82%
소프트웨어 컴포넌트 테스트 보고서	8	73%
소프트웨어 소스코드 검증 보고서	0	0%

- 항공분야

구현 단계의 산출물이 테스트에 필요한 정보를 충분히 제공하고 있는지에 대한 공통 질문에서 항공 분야의 경우 “그렇다, 매우 그렇다” 가 45.5%, “약간 그렇다” 가 36.4%로 파악되었다.

세부적으로 “구현 단계에서 작성하는 산출물” 을 선택하라는 질문에 항공 분야는 “소프트웨어 소스코드 및 지원문서” 라는 응답이 가장 많았다.

<표 4-126> 구현 단계에서 작성하는 산출물

산출물	건수	%
소프트웨어 소스코드 및 지원 문서	13	100%
소프트웨어 단위 테스트 보고서	8	62%
소프트웨어 소스코드 검증 보고서	8	62%

4) 테스트 단계

13. 구현된 소프트웨어에 대한 단위/통합 테스트가 충분히 이루어지고 있다고 생각하십니까?

1. 전혀 아니다 2. 아니다 3. 약간 아니다 4. 약간 그렇다 5. 그렇다 6.매우 그렇다

<표 4-127> 구현된 소프트웨어 대해 테스트가 충분한지에 대한 현황 분석

(단위 : 건수, 백분율)

산업분야	전혀 아니다		아니다		약간 아니다		약간 그렇다		그렇다		매우 그렇다	
비표준	1	6.7	2	13.3	2	13.3	3	20.0	7	46.7	-	0.0
자동차	1	9.1	1	9.1	2	18.2	3	27.3	4	36.4	-	0.0
철도	-	0.0	2	18.2	-	0.0	2	18.2	6	54.5	1	9.1
항공	-	0.0	1	7.7	-	0.0	4	30.8	7	53.8	1	7.7
표준합계	1	2.9	4	11.4	2	5.7	9	25.7	17	48.6	2	5.7

- 비표준 분야

구현된 소프트웨어에 대한 단위/통합 테스트가 충분히 이루어지고 있는지에 대한 공통 질문에서, 비표준 분야에서는 “그렇다” 가 46.7%, “약간 그렇다” 가 20%로 조사되었다. 세부적으로 “소프트웨어 테스트를 위해 수행하는 활동” 을 선택하라는 질문에 비표준 분야는 “개발된 소프트웨어의 의도된 기능이 정상적으로 작동 되는지를 적절히 시험함” 이라는 응답이 가장 많이 조사되었다.

<표 4-128> 비표준 분야 소프트웨어 테스트를 위해 수행하는 활동

활동	건수	%
개발된 소프트웨어가 설계 단계에서 계획/명세한 대로 테스트하고 있음.	7	47%
개발된 소프트웨어의 의도된 기능이 정상적으로 작동 되는지를 적절히 시험함.	11	73%
모든 소프트웨어 모듈/통합/시스템 시험 결과를 기록함	6	40%
시험 후 실패된 내용에 대한 수정 및 검토 활동을 공식적으로 수행함	4	27%

“소프트웨어 안전기능을 확보를 위한 테스트가 충분히 이루어지고 있다고 생각하십니까?” 라는 질문에 “약간 그렇다” 가 5건, “그렇다” 가 4건이었으며, “아니다” 또한 4건

이었다. “약간 아니다” 와 “전혀 아니다” 는 각각 1건이었다.

- 자동차 분야

구현된 소프트웨어에 대한 단위/통합 테스트가 충분히 이루어지고 있는지에 대한 공통 질문에서, 자동차 분야의 경우 “그렇다” 가 36.4%, “약간 그렇다” 가 27.3%로 조사되었다. 세부적으로 “SW안전성 확보를 위한 통합 및 시험 활동” 을 선택하라는 질문에 자동차 분야는 “설정 데이터 검증” 이 가장 많은 것으로 조사되었다.

<표 4-129> 자동차 분야 SW안전성 확보를 위한 통합 및 시험 활동

활동	건수	%
단계별 기능안전 요건 시험	6	55%
설정 데이터 검증	8	73%
보정 데이터 검증 (보정데이터 유효값, 보정데이터 의도/용도, 범위, 상호의존성)	6	55%

“소프트웨어 통합 시험을 위한 방법에 대해 적용하고 있는 방법” 에 대한 질문에서 자동차 분야는 “요구사항 기반 시험” 이라는 응답이 가장 많았다.

<표 4-130> 소프트웨어 통합 시험을 위한 방법에 대해 적용하고 있는 방법

활동	건수	%
요구사항기반시험	11	100%
인터페이스 시험	8	73%
결함 주입 시험 (예.변수의 변수 값 손상, 변이 코드 또는 CPU 레지스터의 값 손상등 주입)	5	45%
자원 사용 시험 (단, 에뮬레이터가 자원 사용 시험을 지원하는 경우에만 사용)	2	18%
모델과 코드 간 비교 시험(back-to-back test) (시뮬레이션할 수 있는 모델이 필요)	2	18%
자동화된 툴 활용	0	0%

- 철도 분야

구현된 소프트웨어에 대한 단위/통합 테스트가 충분히 이루어지고 있는지에 대한 공통 질문에서 철도분야는 “그렇다, 매우 그렇다” 가 63.6%, “약간 그렇다” 가 18.2%로 조사되었다.

세부적으로 “소프트웨어 통합을 위해 수행하는 활동” 을 선택하라는 질문에 철도 분야는

“통합은 개별 컴포넌트 및 컴포넌트 인터페이스, 기 테스트된 컴포넌트를 포함하여 점진적 수행” 이라는 응답이 가장 많았다.

<표 4-131> 철도 분야 소프트웨어 통합을 위해 수행하는 활동

활동	건수	%
통합은 개별 컴포넌트 및 컴포넌트 인터페이스, 기 테스트된 컴포넌트를 포함하여 점진적 수행	9	82%
통합 시 변경이 발생하면 영향분석 후 관련 컴포넌트를 식별하고 재검증 활동 수행	6	55%
소프트웨어 통합 보고서는 테스트 보고서의 형식을 준수하여 작성	7	64%
소프트웨어 통합 테스트 보고서는 기법을 올바르게 선택/사용하였는지 여부 증명	5	45%

철도 분야에서는 “소프트웨어/하드웨어 통합을 위해 수행하는 활동”에 대한 분석으로 “소프트웨어/하드웨어 통합 테스트 보고서는 테스트 보고서의 형식을 준수하여 작성” 한다는 응답이 가장 많았다.

<표 4-132> 철도 분야 소프트웨어/하드웨어 통합을 위해 수행하는 활동

활동	건수	%
통합시 변경이 발생하면 영향분석 후 관련 컴포넌트를 식별하고 재검증 활동 수행	6	55%
소프트웨어/하드웨어 통합 테스트 보고서는 테스트 보고서의 형식을 준수하여 작성	10	91%
소프트웨어/하드웨어 통합 테스트 보고서는 기법을 올바르게 선택/사용하였는지 여부 증명	6	55%

추가적인 활동으로 “통합 검증을 위해 수행하는 활동”에 대한 조사로 “소프트웨어 통합 검증 보고서는 검증자(Verifier)의 책임 하에 작성”, “소프트웨어 통합 검증 보고서는 검증 보고서의 형식을 준수하여 작성”, “소프트웨어 통합 테스트 보고서와 소프트웨어·하드웨어 통합 테스트 보고서 검증내용을 작성” 한다는 응답이 가장 많았다.

<표 4-133> 철도 분야 통합 검증을 위해 수행하는 활동

활동	건수	%
소프트웨어 통합 검증 보고서는 검증자(Verifier)의 책임 하에 작성한다.	7	64%
소프트웨어 통합 검증 보고서는 검증 보고서의 형식을 준수하여 작성	7	64%
소프트웨어 통합 테스트 보고서와 소프트웨어/하드웨어 통합 테스트 보고서 검증내용 작성	7	64%

“종합 소프트웨어 테스트를 위해 수행하는 활동” 에 대한 질문에서는 “종합 소프트웨어 테스트 보고서는 테스트 보고서의 형식을 준수하여 작성” 한다는 활동이 가장 많은 것으로 조사되었다.

<표 4-134> 철도 분야 종합 소프트웨어 테스트를 위해 수행하는 활동

활동	건수	%
종합 소프트웨어 테스트 보고서는 테스터(Tester)의 책임 하에 작성	6	55%
종합 소프트웨어 테스트 보고서는 테스트 보고서의 형식을 준수하여 작성	10	91%
확인자(Validator)는 재량에 따라 특정한 추가 테스트를 정의하고 테스트 수행	3	27%
실제 사용자의 필요를 반영한 복잡한 시나리오로 부하 테스트도 수행	5	45%
모든 테스트와 분석 결과들은 종합 소프트웨어 테스트 보고서에 기록	9	82%
실 하드웨어 연결, 운영 인터페이스로 실 시스템 연결, 입출력 신호의 시뮬레이션으로 테스트	7	64%
시스템에 요구되는 일반 운영모드 및 비정상 조건들에 대해서도 테스트	5	45%
시뮬레이션 입출력 데이터가 사용된 경우 실 입출력 데이터와 다르지 않음을 증명	5	45%

“소프트웨어 확인을 위해 수행하는 활동” 은 “소프트웨어 확인 보고서는 확인 보고서를 작성하기 위한 일반 요구사항을 준수하여 작성” 이 가장 많았다.

<표 4-135> 소프트웨어 확인을 위해 수행하는 활동

활동	건수	%
소프트웨어 확인 보고서는 소프트웨어 확인 계획에 근거하여 확인자(Validator)의 책임 하에 작성	7	64%
소프트웨어 확인 보고서는 확인 보고서를 작성하기 위한 일반 요구사항을 준수하여 작성	10	91%
소프트웨어 확인 보고서는 각 기법 및 대책의 조합에 대한 검토내용 포함	7	64%
발견된 결함, 불일치 사항들은 별도 장절에 식별	3	27%

- 항공분야

구현된 소프트웨어에 대한 단위/통합 테스트가 충분히 이루어지고 있는지에 대한 공통 질문에서 항공 분야의 경우 “그렇다, 매우 그렇다”가 54.6%, “약간 그렇다”가 45.5%로 파악되었다. 세부적으로 “SW안전성 확보를 위한 소프트웨어 테스트를 수행하는 환경”을 선택하라는 질문에 항공 분야는 “대상 컴퓨터에 로드된 소프트웨어가 포함된 테스트 환경”이라는 응답이 가장 많았다.

<표 4-136> SW안전성 확보를 위한 소프트웨어 테스트를 수행하는 환경

수행 환경	건수	%
대상 컴퓨터에 로드 된 소프트웨어가 포함된 테스트 환경	10	77%
대상 컴퓨터 환경과 매우 유사한 환경	4	31%
테스트 환경은 고려하지 않음.	1	8%

또한, 항공분야에서 “SW안전성 확보를 위한 테스트”에 대한 질문에서는 “요구사항 기반 하드웨어・소프트웨어 통합 테스트”라는 응답이 가장 조사되었다.

<표 4-136> 항공 분야 SW안전성 확보를 위한 테스트

활동	건수	%
요구사항 기반 low-level 테스트	9	69%
요구사항 기반 소프트웨어 통합테스트	11	85%
요구 사항 기반 하드웨어 / 소프트웨어 통합 테스트	13	100%

비표준 분야의 테스트는 설계단계의 추적성 수행이 부족하며, 자동차 부분은 안전 활동에 주력하기 보다는 기본적인 소프트웨어 개발활동에 맞추어 테스트를 진행한다. 철도의 경우는 개별 컴포넌트 및 컴포넌트 인터페이스, 기 테스트된 컴포넌트를 포함하여 점진적 수행하고 있었다.

산업 분야별로 “테스트가 충분히 이루어지지 않고 있는 이유”에 대한 조사 결과 분야별로 다음과 같은 의견이 조사되었다.

〈표 4-137〉 테스트가 충분히 이루어지지 않고 있는 이유 분석

분야	응답
비표준	• 개발시간 부족, 인력 부족, 비용 부족, 인식 부족 • SW공학 모름, 테스트 명세가 적절히 이루어지지 않음
자동차	• Test 환경의 제약, 시간 및 비용부족 • 임베디드 개발 업무에서는 SW 안전성에 대한 인식 부족 • 단위 테스트와의 경계 모호 • 환경을 갖추기 어려움, 시험수행 담당자 지정/육성 어려움
철도	• 인력부족, 개발일정에 의한 기간부족이 많음 • 전문지식 부족, 인력, 시간, 예산(종합시뮬레이션 제작)
항공	• 인력, 예산부족 • 기능, 성능 구현 위주의 구조설계만 가능 • 최종 타켓이 프로젝트 말미에 나오게 되는 시간적 한계

14. 테스트 단계에서 안전성 보증을 위한 기법을 충분히 적용하고 있다고 생각하십니까?

1. 전혀 아니다 2. 아니다 3. 약간 아니다 4. 약간 그렇다 5. 그렇다 6.매우 그렇다

〈표 4-138〉 테스트 단계에서 안전성 보증 기법이 어느 정도 적용되는지 현황 분석

(단위 : 건수, 백분율)

산업분야	전혀 아니다		아니다		약간 아니다		약간 그렇다		그렇다		매우 그렇다	
비표준	1	6.7	3	20.0	3	20.0	7	46.7	1	6.7	-	0.0
자동차	1	9.1	1	9.1	2	18.2	5	45.5	2	18.2	-	0.0
철도	-	0.0	1	9.1	1	9.1	2	18.2	6	54.5	1	9.1
항공	-	0.0	3	23.1	1	7.7	5	38.5	3	23.1	1	7.7
표준합계	1	2.9	5	14.3	4	11.4	12	34.3	11	31.4	2	5.7

- 비표준 분야

테스트 단계에서 안전성 보증을 위한 기법을 충분히 적용하고 있는지에 대한 공통 질문에

서, 비표준 분야에서는 “그렇다” 가 6.7%, “약간 그렇다” 가 46.7%로 조사되었다. 세부적으로 “SW 안전성 확보를 위해 모듈/통합 테스트 기법” 을 선택하라는 질문에 비표준 분야는 “성능 및 인터페이스 테스트” 라는 응답이 가장 많이 조사되었다.

<표 4-139> 비표준 분야 SW 안전성 확보를 위해 모듈/통합 테스트 기법

기법	건수	%
동적 분석 및 테스트 (커버리지 테스트)	4	27%
기능 및 블랙 박스 테스트	6	40%
성능 및 인터페이스 테스트	11	73%
테스트 관리 및 자동화 도구 사용	4	27%
소프트웨어 안전 요구 사항, 시험 계획 간 추적	5	33%

- 자동차 분야

테스트 단계에서 안전성 보증을 위한 기법을 충분히 적용하고 있는지에 대한 공통 질문에서, 자동차 분야의 경우 “그렇다” 가 18.2%, “약간 그렇다” 가 45.5%로 조사되었다.

세부적으로 “SW 안전성 확보를 위해 모듈/통합 테스트 기법” 을 선택하라는 질문에 자동차 분야는 “기능 및 블랙박스 테스트” 와 “성능 테스트” 가 가장 많은 것으로 조사되었다.

<표 4-140> 자동차 분야 SW 안전성 확보를 위해 모듈/통합 테스트 기법

기법	건수	%
기능 및 블랙 박스 테스트	9	82%
성능 테스트	9	82%
HW와 SW 통합 테스트	5	45%
시스템과 SW 설계 요구사항 간의 추적성 테스트	3	27%

“소프트웨어 안전 요구사항 검증 실시를 위한 환경 구현 방법” 에 대한 질문에서 자동차 분야는 “임베디드 소프트웨어가 소프트웨어 안전 요구사항 충족 조건 검증” 이라는 응답이 가장 많았다.

<표 4-141> 소프트웨어 안전 요구사항 검증 실시를 위한 환경 구현 방법

환경 구현 방법	건수	%
임베디드 소프트웨가 소프트웨어 안전 요구사항 충족 조건 검증	7	64%
루프내의 하드웨어 검증	4	36%
전자 제어 단위 네트워크 환경 검증	3	27%
차량 직접 검증	6	55%

- 철도 분야

테스트 단계에서 안전성 보증을 위한 기법을 충분히 적용하고 있는지에 대한 공통 질문에서 철도분야는 “그렇다, 매우 그렇다”가 63.6%, “약간 그렇다”가 18.2%로 조사되었다. 세부적으로 “소프트웨어 통합을 위해 적용한 기능 및 블랙박스 테스팅, 성능 테스팅 대책 및 기법”을 선택하라는 질문에 철도 분야는 “경계값 분석 (Boundary Value Analysis)”이라는 응답이 가장 많았다.

<표 4-142> 소프트웨어 통합을 위해 적용한 기능, 테스팅 대책 및 기법

활동	건수	%
경계값 분석 (Boundary Value Analysis)	10	91%
동등 분할 테스팅 (Equivalence Classes and Inp ut Partition Testing)	6	55%

“소프트웨어/하드웨어 통합을 위해 적용한 기능 및 블랙박스 테스팅, 성능 테스팅 대책 및 기법” 또한 “경계값 분석 (Boundary Value Analysis)”이 가장 많았다.

<표 4-143> 소프트웨어/하드웨어 통합을 위해 적용한 기능, 테스팅 대책 및 기법

활동	건수	%
경계값 분석 (Boundary Value Analysis)	10	91%
동등 분할 테스팅 (Equivalence Classes and Inp ut Partition Testing)	5	45%

- 항공분야

테스트 단계에서 안전성 보증을 위한 기법을 충분히 적용하고 있는지에 대한 공통 질문에서 항공 분야의 경우 “그렇다, 매우 그렇다”가 36.4%, “약간 그렇다”가 27.3%로 파악되었다.

세부적으로 “수행하는 테스트 기법”을 선택하라는 질문에 항공 분야는 “Normal test 기법”이 가장 많았다.

<표 4-144> 수행하는 테스트 기법

테스트 수행 기법	건수	%
Normal test 기법	11	85%
Robustness test 기법	9	69%

“Normal test 기법 중 확인하는 내용”에 대한 질문에서는 “상태 전이(state transitions)의 경우 정상적인 시스템 운영 동안 발생될 수 있는 모든 상태 전이에 대한 테스트 케이스를 개발”한다는 응답이 가장 조사되었다.

<표 4-145> Normal test 기법 중 확인하는 내용

Normal Test 기법	건수	%
실수 및 정수 입력 변수에 대한 valid equivalence classes 과 boundary values를 사용	9	69%
시간 관련 함수(예:filters, integrators, and delays)의 경우 코드의 반복 수행	8	62%
상태전이(state transitions)의 경우 정상적인 시스템 운영 동안 발생 될 수 있는 모든 상태전이에 대한 테스트 케이스 개발	10	77%
논리 방정식으로 표현 된 소프트웨어 요구 사항의 경우 변수 사용 및 부울 연산자 검증	9	69%

“Robustness test 기법 중 확인하는 내용”은 “예외적인 입력 값에 대한 오류 모드(failure mode)를 결정”이 가장 많았다.

<표 4-146> Robustness test 기법 중 확인하는 내용

Robustness test 기법	건수	%
실수 및 정수 입력 변수에 대한 equivalence class selection of invalid values를 사용	7	54%
비정상적인 조건에서 시스템을 초기화	9	69%
예외적인 입력 값에 대한 오류 모드(failure mode)를 결정	10	77%
산술적 오버플로우 조건에 대하여 테스트	6	46%

“테스트 커버리지 분석을 수행하는 범위”는 “상위수준 요구사항 기반 테스트 커버리지 분석”이라는 응답이 가장 많았다.

<표 4-147> 항공 분야 테스트 커버리지 분석을 수행하는 범위

분석 범위	건수	%
상위 수준 요구사항 기반 테스트 커버리지 분석	10	77%
상세 요구사항 기반 테스트 커버리지 분석	7	54%
구조적 커버리지 분석	6	46%

산업 분야를 종합해보면 철도가 가장 수준이 높았으며 항공, 자동차 순으로 높은 수준을 보였다. 비표준 분야는 타 분야에 비해 수준이 상대적으로 낮았다. 평균적으로는 “그렇다, 매우 그렇다”가 31.2%, “약간 그렇다”가 34.4%로 나타났다.

15. 테스트 단계의 산출물이 계획했던 SW 안전성 요구사항을 완료하였음을 충분히 제시하고 있습니까?

1. 전혀 아니다 2. 아니다 3. 약간 아니다 4. 약간 그렇다 5. 그렇다 6.매우 그렇다

<표 4-148> 테스트 단계 산출물이 계획했던 요구사항을 충족 여부에 대한 현황 분석

(단위 : 백분율)

산업분야	전혀 아니다		아니다		약간 아니다		약간 그렇다		그렇다		매우 그렇다	
비표준	1	6.7	4	26.7	1	6.7	5	33.3	4	26.7	-	0.0
자동차	-	0.0	-	0.0	3	27.3	4	36.4	4	36.4	-	0.0
철도	-	0.0	-	0.0	1	9.1	3	27.3	6	54.5	1	9.1
항공	1	9.1	1	9.1	-	0.0	4	36.4	4	36.4	1	9.1
합계	1	3.0	1	3.0	4	12.1	11	33.3	14	42.4	2	6.1

- 비표준 분야

테스트 단계의 산출물이 계획했던 소프트웨어 안전성 요구사항을 완료하였음을 충분히 제시하고 있는지에 대한 공통 질문에서, 비표준 분야에서는 “그렇다”가 26.7%, “약간 그렇다”가 33.3%로 조사되었다.

세부적으로 “통합 시험 명세에 포함되는 것”을 선택하라는 질문에 비표준 분야는 “시스템을 통합 단계에 따라 구분”과 “테스트 케이스와 시험 데이터, 시험의 형태”라는 응답이 가장 많이 조사되었다.

<표 4-149> 비표준 분야 통합 시험 명세에 포함되는 것

명세서 포함 내용	건수	%
시스템을 통합 단계에 따라 구분	7	47%
테스트 케이스와 시험 데이터, 시험의 형태	7	47%
도구, 지원 소프트웨어, 구성 설명 등의 시험 환경, 완료 판단기준	3	20%

- 자동차 분야

테스트 단계의 산출물이 계획했던 소프트웨어 안전성 요구사항을 완료하였음을 충분히 제시하고 있는지에 대한 공통 질문에서, 자동차 분야의 경우 “그렇다”가 36.4%, “약간 그렇다”가 36.4%로 조사되었다.

세부적으로 “테스트 단계 수행관련 산출물 보유 및 사용하는 산출물”을 선택하라는 질문에 자동차 분야는 “소프트웨어 검증 명세서(갱신)” 작성이 가장 많은 것으로 조사되었다.

<표 4-150> 자동차 분야 테스트 단계 수행관련 산출물 보유 및 사용하는 산출물

산출물	건수	%
소프트웨어 검증 계획 (갱신)	10	91%
소프트웨어 검증 명세서 (갱신)	11	100%
임베디드 소프트웨어	4	36%
소프트웨어 검증 보고서 (갱신)	9	82%

- 철도 분야

테스트 단계의 산출물이 계획했던 소프트웨어 안전성 요구사항을 완료하였음을 충분히 제시하고 있는지에 대한 공통 질문에서 철도분야는 “그렇다, 매우 그렇다”가 63.6%, “약간 그렇다”가 27.3%로 조사되었다.

세부적으로 “테스트 단계 수행관련 산출물 보유 및 사용하는 산출물”을 선택하라는 질문에 철도 분야는 “소프트웨어 통합 테스트 결과서”라는 응답이 가장 많았다.

<표 4-151> 철도 분야 테스트 단계 수행관련 산출물 보유 및 사용하는 산출물

산출물	건수	%
소프트웨어 통합 테스트 결과서	10	91%
소프트웨어/하드웨어 통합 테스트 결과서	9	82%
소프트웨어 통합 검증 보고서	8	73%

- 항공분야

테스트 단계의 산출물이 계획했던 소프트웨어 안전성 요구사항을 완료하였음을 충분히 제시하고 있는지에 대한 공통 질문에서 항공 분야의 경우 “그렇다, 매우 그렇다”가 33.3%, “약간 그렇다”가 22.2%로 파악되었다.

세부적으로 “테스트 단계 수행관련 산출물 보유 및 사용하는 산출물”을 선택하라는 질문에 항공 분야는 “SW 검증 결과(Software Verification Results)”라는 응답이 가장 많았다. 기타 의견으로 방위사업청의 규정 문서를 작성, 문제 보고서를 작성한다는 의견도 조사되었다.

<표 4-152> 항공 분야 테스트 단계 수행관련 산출물 보유 및 사용하는 산출물

산출물	건수	%
SW 검증계획서(Software Verification Plan)	10	77%
SW 검증 사례 및 절차(Software Verification Cases and Procedures)	9	69%
SW 검증 결과(Software Verification Results)	11	85%

산업 분야를 종합해보면 철도가 가장 수준이 높았으며 자동차, 항공 순으로 높은 수준을 보였다. 비표준 분야는 타 분야에 비해 수준이 상대적으로 낮았다. 평균적으로는 “그렇다, 매우 그렇다”가 40%, “약간 그렇다”가 35.4%로 나타났다.

4. 소프트웨어 안전 확산 실태 조사 분석

SW 제품 안전 확보를 위해 SW 안전 개발 프로세스에 대한 일반적인 인식과 SW 안전 개발 프로세스 확산을 위해 무엇을 지원해야 할 것인지 조사하기 위해 소프트웨어 안전 개발 프로세스에 대한 지켜지지 않은 이유와 효과적인 수행을 위한 개선 방안을 파악하고 분석한다. 또한, 소프트웨어 안전 개발 프로세스에 대한 인식을 조사하고 분석한다.

1) SW안전 일반 현황

SW안전 개발 프로세스 각 단계 활동이 잘 지켜지지 않은 이유는 무엇입니까?

1. 해당 활동이 필요 없다고 생각함
2.어떻게 해야 할지에 대한 담당자의 지식이 부족
3.제대로 수행하기 위한 적당한 도구가 없음
4.최신 기술을 사용해서 이에 대한 구현 방안 없음
5.해당 활동을 수행할 비용 또는 시간이 부족함
6.안전 개발 프로세스보다 제품 개발 효율성을 추구
7. 기타

<표 4-153> 소프트웨어 안전 개발 프로세스 각 단계 활동이 잘 안 되는 이유 분석

(단위 : 건수, 백분율)

구분	비표준		자동차		철도		항공		표준합계	
해당 활동이 필요 없다고 생각함	3	20	1	9	4	36	9	69	14	40
어떻게 해야 할지에 대한 담당자의 지식이 부족	10	67	8	73	2	18	5	38	15	43
제대로 수행하기 위한 적당한 도구가 없음	4	27	3	27	0	0	0	0	3	9
최신 기술을 사용해서 이의 구현 방안이 없음	3	20	1	9	9	82	7	54	17	49
해당 활동을 수행할 비용 또는 시간이 부족함	13	87	7	64	6	55	8	62	21	60
안전 개발 프로세스보다 제품 개발 효율성을 추구	6	40	4	36	2	18	0	0	6	17
기타	0	0	2	18	0	0	0	0	2	6

* 중복 응답, 응답기업수는 <표4-1> 참조, 백분율(%) 계산방식 = 응답수/각부분별 응답기업수

- 비표준 분야

비표준 분야에서는 소프트웨어 안전 개발 프로세스 각 단계 활동이 안 되는 이유로는 “해당 활동을 수행할 비용, 시간 부족”, “해당 활동이 필요 없다고 행각함”, “안전 개발 프로세스보다 제품 개발 효율성 추가” 순으로 조사되었다.

- 자동차

자동차 분야에서는 소프트웨어 안전 개발 프로세스 각 단계 활동이 안 되는 이유로는 “해당 활동이 필요 없다고 행각함”, “해당 활동을 수행할 비용, 시간 부족”, “안전 개발 프로세스보다 제품 개발 효율성 추가” 순으로 조사되었다. 기타 의견으로는 고객의 요구사항이 없거나 현재 인력 충원을 검토 중이라는 의견도 조사되었다.

- 철도

철도 분야에서는 소프트웨어 안전 개발 프로세스 각 단계 활동이 안 되는 이유로는 “제도로 수행하기 위한 적당한 도구가 없음”, “해당 활동을 수행할 비용, 시간 부족” 순으로 조사되었다.

- 항공

항공 분야에서는 소프트웨어 안전 개발 프로세스 각 단계 활동이 안 되는 이유로는 “해당 활동을 수행할 비용, 시간 부족”, “최신 기술을 사용하여 구현 방안이 없음”, “해당 활동이 필요 없다고 행각함” 순으로 조사되었다.

산업 분야를 종합해보면 각 분야별로 담당자의 지식 부족 문제와 비용 및 시간의 부족 문제를 제시했다. 이는 각각의 단계에서 각각의 단계가 제대로 수행되지 못한 이유와 동일하다. 특이하게 철도의 경우는 최신 기술을 사용하기 때문에 구현 방안이 없다는 응답이 많았다. 이는 철도도 현재 무인 기술이 들어오고, 안전을 위해 이중화 문제를 해결해야 하는 데 이에 대한 대응 방안이 없는 것이 문제인 것으로 파악되었다. 항공의 경우는 해당 활동이 필요 없다고 생각한다는 의견이 많았는데, 이는 안전 활동을 하지 않는 다는 부정적인 대답이라기보다는 필요가 없는 부분은 안전 활동을 하지 않고 있다고 긍정적으로 해석할 수 있겠다. 이는 다른 질문을 통해 항공은 위험분석 및 안전 매커니즘 구현을 다른 부분보다는 정확하게 수행하고 있다는 사실이 조사되었기 때문이다.

SW 안전 개발 프로세스의 단계별 활동을 잘 수행하기 위한 개선 방안은 무엇입니까?

1. 참여 인력의 안전 관련 의식 제고
2. 안전 프로세스, 기법, 산출물 등에 대한 충분한 교육 제공
3. 효과적인 관리 및 검증 도구 제공
4. 전문기관, 학회 등을 통한 SW 안전 기술 연구
5. 안전성 관리 및 보증을 위한 비용 지원
6. 제품 개발 안전과 효율성과의 객관적 판단기준 마련

<표 4-154> SW안전 개발 프로세스 효과적으로 수행하기 위한 개선 방안

(단위 : 건수, 백분율)

구분	비표준		자동차		철도		항공		표준합계	
참여 인력의 안전 관련 의식 제고	10	67	9	82	6	55	10	77	25	71
안전 프로세스, 기법, 산출물 등 충분한 교육 제공	9	60	10	91	5	45	7	54	22	63
효과적인 관리 및 검증 도구 제공	6	40	5	45	1	9	7	15	8	23
전문기관, 학회 등을 통한 SW 안전 기술 연구	3	20	3	27	9	82	9	69	21	60
안전성 관리 및 보증을 위한 비용 지원	8	53	7	64	3	27	7	54	17	49
제품 개발 안전과 효율성과의 객관적 판단기준 마련	8	53	5	45	0	0	0	0	5	14

* 중복 응답, 응답기업수는 <표4-1> 참조, 백분율(%) 계산방식 = 응답수/각부분별 응답기업수

- 비표준 분야

비표준 분야에서는 소프트웨어 안전 개발 프로세스를 효과적으로 수행하기 위한 개선 방안으로는 “참여 인력의 안전 관련 인식 제고”, “안전 프로세스, 기법, 산출물 등에 충분한 교육 제공”, “안전성 관리 및 보증을 위한 비용 지원” 순으로 조사되었다.

- 자동차

자동차 분야에서는 소프트웨어 안전 개발 프로세스를 효과적으로 수행하기 위한 개선 방안으로는 “참여 인력의 안전 관련 인식 제고”, “안전 프로세스, 기법, 산출물 등에 충분한 교육 제공”, “안전성 관리 및 보증을 위한 비용 지원” 순으로 조사되었다.

- 철도

철도 분야에서는 소프트웨어 안전 개발 프로세스를 효과적으로 수행하기 위한 개선 방안으로는 “전문기관, 학회 등을 통한 SW 안전 기술 연구”, “참여 인력의 안전 관련 인식 제고”, “안전 프로세스, 기법, 산출물 등에 충분한 교육 제공” 순으로 조사되었다.

- 항공

항공 분야에서는 소프트웨어 안전 개발 프로세스를 효과적으로 수행하기 위한 개선 방안으로는 “전문기관, 학회 등을 통한 SW 안전 기술 연구”, “참여 인력의 안전 관련 인식 제고”, “안전 프로세스, 기법, 산출물 등에 충분한 교육 제공” 순으로 조사되었다.

산업 분야를 종합해보면 소프트웨어 개발 프로세스를 효과적으로 수행하기 위한 방안으로 비표준 분야에서는 참여 인력의 안전 관련 의식 제고가 필요, 자동차 분야에서는 안전 프로세스, 기법, 산출물에 대한 충분한 교육 필요, 철도에서는 전문 기관, 학회 등을 통한 소프트웨어 안전 기술 연구, 항공 분야에서는 참여 인력의 안전 관련 인식이나 전문기관, 학회 등 소프트에어 안전 기술 연구 등을 통해 개선을 해야 한다는 의견을 제시하였다.

공통적으로는 참여인력의 안전 관련 의식 제고 및 안전 프로세스, 기법, 산출물 등 충분한 교육 제공에 대한 요구가 많았다. 철도 분야는 문제점 부분에서 최신기술에 대한 안전 구현 방안이 없다고 했으므로, 전문기관, 학회 등을 통한 SW안전 기술 연구를 최우선 개선방안으로 제시하였다.

2) 소프트웨어 안전 개발 프로세스 인식 분석

귀사는 향후 SW안전 확보가 중요하다고 생각하십니까?
1. 전혀아니다 2. 아니다 3. 약간 아니다 4. 약간 그렇다 5. 그렇다 6.매우 그렇다

<표 4-155> 향후 소프트웨어 안전성 확보 중요성 인식 분석

(단위 : 건수, 백분율)

산업분야	전혀아니다		아니다		약간아니다		약간그렇다		그렇다		매우그렇다	
비표준	0	0	0	0	0	0	2	13	9	60	4	27
자동차	0	0	0	0	0	0	1	9	7	64	3	27
철도	0	0	0	0	0	0	0	0	2	18	9	82
항공	0	0	0	0	0	0	0	0	6	46	7	54
표준합계	0	0	0	0	0	0	1	3	15	43	19	54

- 비표준 분야

비표준 분야에서는 소프트웨어 안전이 향후 귀사의 업무에 중요하다고 판단되는가에 대한 조사로 “그렇다, 매우 그렇다” 가 87%로 조사되었고, “약간 그렇다” 가 13%로 조사되었다.

- 자동차

자동차 분야에서는 소프트웨어 안전이 향후 귀사의 업무에 중요하다고 판단되는가에 대한 조사로 “그렇다, 매우 그렇다” 가 91%로 조사되었고, “약간 그렇다” 가 9%로 조사되었다.

- 철도

철도 분야에서는 소프트웨어 안전이 향후 귀사의 업무에 중요하다고 판단되는가에 대한 조사로 “매우 그렇다” 82%, “그렇다” 가 18%로 조사되었다. 기타 의견으로 소프트웨어 개발 일정 확보 및 비용 확보라는 의견도 조사되었다.

- 항공

항공 분야에서는 소프트웨어 안전이 향후 귀사의 업무에 중요하다고 판단되는가에 대한 조사로 “매우 그렇다” 54%, “그렇다” 가 46%로 조사되었다.

SW안전 확보를 위해 중요하다고 생각되는 항목은 무엇입니까?

1. 안전 개발에 대한 회사 인식	2. 안전 개발 관련 지식이나 정보	3. 안전 개발 프로세스
4. 요구사항 명세	5. 코딩 표준	6. 검증 및 테스팅
7. 위험도 분석	8. 인증	9. 기타

<표 4-156> 소프트웨어 안전 확보를 위해 중요하다고 생각되는 항목 분석

(단위 : 건수, 백분율)

구분	비표준		자동차		철도		항공		표준합계	
안전 개발에 대한 회사 인식	8	53	7	64	5	45	8	62	20	57
안전 개발 관련 지식이나 정보	8	53	7	64	8	73	9	69	24	69
안전 개발 프로세스	8	53	7	64	3	27	5	38	15	43
요구사항 명세	5	33	5	45	3	27	3	23	11	31
코딩 표준	6	40	3	27	5	45	5	38	13	37
검증 및 테스팅	8	53	4	36	4	36	6	46	14	40
위험도 분석	9	60	4	36	3	27	7	54	14	40
인증	2	13	1	9	1	9	0	0	2	6

* 중복 응답, 응답기업수는 <표4-1> 참조, 백분율(%) 계산방식 = 응답수/각부분별 응답기업수

- 비표준 분야

비표준 분야에서는 소프트웨어 안전을 위해 중요하다고 생각되는 항목에 대한 조사에서는 “위험도 분석”, “안전 개발 인식”, “안전 개발 관련 지식이나 정보”, “안전 개발 프로세스” 순으로 조사되었다.

- 자동차

자동차 분야에서는 소프트웨어 안전을 위해 중요하다고 생각되는 항목에 대한 조사에서는 “안전 개발 인식”, “안전 개발 관련 지식이나 정보”, “안전 개발 프로세스” 순으로 조사되었다.

- 철도

철도 분야에서는 소프트웨어 안전을 위해 중요하다고 생각되는 항목에 대한 조사에서는 “안전 개발 관련 지식 및 정보”, “안전 개발 인식”, “코딩 표준” 순으로 조사되었다. 기타 의견으로 소프트웨어 개발 일정 확보 및 비용 확보라는 의견도 조사되었다.

- 항공

항공 분야에서는 소프트웨어 안전을 위해 중요하다고 생각되는 항목에 대한 조사에서는 “안

전 개발 관련 지식이나 정보”, “개발 인식”, “안전 개발 프로세스” 순으로 조사되었다.

산업 분야를 종합 분석하면 소프트웨어 안전성 확보에 중요하다고 생각되는 항목 조사에서 비표준 분야에서는 위험도 분석, 검증 및 테스팅, 안전 개발 인식, 안전 개발 관련 지식 및 정보, 안전 개발 프로세스가 중요한 항목이라고 조사되었고, 자동차 분야에서는 안전 개발 인식, 안전 개발 지식 및 정보, 안전 개발 프로세스가 상대적으로 중요하다고 조사되었다. 철도 분야는 안전 개발 지식 및 정보, 안전 개발 인식, 코칭 표준 등이 중요한 항목이라고 조사되었고, 항공 분야는 안전 개발 관련 지식 및 정보, 안전 개발 인식이 중요한 항목으로 조사되었다.

각 분야별로 공통적으로 중요한 항목으로 생각되는 부분은 안전 개발 인식 및 안전 개발 지식 및 정보로 조사되었다. 즉 각각의 기법이나 활동보다는 안전 확보를 위한 근본적인 인식 개선과 지식과 정보들이 중요하다고 생각하는 것으로 조사되었다.

SW안전에 대한 관련 정보를 어떠한 경로로 취득하십니까?		
1. 세미나	2. 교육(학교,학원, 온라인 교육)	3. 인터넷 검색
4. 전문가 지원	5. 전문 기업 지원	

〈표 4-157〉 소프트웨어 안전 관련 정보 취득 경로 분석[44)]

(단위 : 건수, 백분율)

구분	비표준		자동차		철도		항공		표준합계	
세미나	9	60	10	91	6	55	6	46	22	64
교육	5	33	7	64	1	9	11	85	19	54
인터넷 검색	5	33	9	82	4	36	3	23	16	46
전문가 지원	7	47	3	27	6	55	0	0	9	26
전문 기업 지원	1	7	2	18	0	0	0	0	2	6

* 중복 응답, 응답기업수는 〈표4-1〉 참조, 백분율(%) 계산방식 = 응답수/각부분별 응답기업수

- 비표준 분야

비표준 분야는 소프트웨어 안전 관련 정보 취득 경로에 대한 조사에서 “세미나”, “전문가 지원”, “교육 이나 인터넷 검색” 순으로 소프트웨어 안전에 대한 정보를 취득하고 있는 것으로 조사되었다.

44) 구분 항목의 백분율 모수는 비표준 15, 자동차 11, 철도 11, 항공 13으로 함

- 자동차

자동차 분야는 소프트웨어 안전 관련 정보 취득 경로에 대한 조사에서 “세미나”, “인터넷 검색”, “교육” 순으로 조사되었다.

- 철도

철도 분야는 소프트웨어 안전 관련 정보 취득 경로에 대한 조사에서 “세미나나 전문가 지원”, “인터넷 검색” 순으로 조사되었다.

- 항공

항공 분야는 소프트웨어 안전 관련 정보 취득 경로에 대한 조사에서 “교육”, “세미나”, “인터넷 검색” 순으로 조사되었다.

산업 분야를 종합 분석하면 소프트웨어 안전 관련 정보 취득 경로에 대한 조사에서 비표준 분야에서는 세미나나 전문가 지원이 많은 것으로 조사되었고, 표준 분야에서는 세미나, 교육, 인터넷 검색 순으로 소프트웨어 안전 관련 정보를 취득하는 것으로 조사되었다.

안전 정보가 아직은 체계적인 교육보다는 단발성 세미나 위주로 취득되는 것으로 조사되었다. 지금까지 조사에서 안전 프로세스 시행이 어려운 이유 중의 하나가 전문가 부족으로 나타났기 때문에, 이를 보완할 수 있는 체계적인 교육이 필요하다 하겠다. 또한 인터넷 검색을 자동차나 항공 등이 안전 정보 취득 수단으로 이용하여, 정보공유 사이트의 제공 또한 필요하다 하겠다.

SW안전 확보를 위해 다음 사항을 답변해주십시오.

1. SW안전 확보를 위한 법제도 개선의 필요성은 어느 정도라고 생각하십니까?
2. SW안전 확보를 위한 정보 공유할 수 있는 시스템의 필요성은 어느 정도라고 생각하십니까?
3. SW안전 확보를 위한 전담 기관 설립 필요성이 어느 정도라고 생각하십니까?
4. SW안전 확보를 위한 전문 인력 양성 필요성이 어느 정도라고 생각하십니까?
5. SW안전 확보를 위한 전문 기업 육성 필요성이 어느 정도라고 생각하십니까?

<표 4-158> 소프트웨어 안전 확보를 위한 방안 분석

(단위 : 백분율)

산업분야		전혀 아니다	아니다	약간 아니다	약간 그렇다	그렇다	매우 그렇다
비표준	법제도 개선 필요성	0%	0%	7%	27%	40%	27%
	정보 공유 시스템 필요성	0%	13%	0%	27%	40%	20%
	전담 기관 설립 필요성	0%	13%	0%	33%	33%	20%
	전문 인력 양성 필요성	0%	7%	0%	27%	40%	27%
	전문 기업 육성 필요성	0%	13%	0%	20%	33%	33%
자동차	법제도 개선 필요성	0%	0%	0%	27%	64%	9%
	정보 공유 시스템 필요성	0%	0%	0%	27%	64%	9%
	전담 기관 설립 필요성	0%	0%	9%	18%	64%	9%
	전문 인력 양성 필요성	0%	0%	0%	0%	82%	18%
	전문 기업 육성 필요성	0%	0%	0%	9%	73%	18%
철도	법제도 개선 필요성	0%	0%	0%	36%	36%	27%
	정보 공유 시스템 필요성	0%	0%	0%	36%	27%	36%
	전담 기관 설립 필요성	0%	0%	0%	9%	55%	36%
	전문 인력 양성 필요성	0%	0%	0%	0%	27%	73%
	전문 기업 육성 필요성	0%	0%	0%	9%	45%	45%
항공	법제도 개선 필요성	0%	0%	15%	23%	31%	31%
	정보 공유 시스템 필요성	0%	0%	0%	15%	38%	46%
	전담 기관 설립 필요성	0%	0%	0%	15%	54%	31%
	전문 인력 양성 필요성	0%	0%	0%	15%	31%	54%
	전문 기업 육성 필요성	0%	0%	0%	23%	31%	46%

- 비표준 분야

비표준 분야에서는 소프트웨어 안전을 위해 중요하다고 생각되는 항목에 대한 조사에서는 “위험도 분석”, “안전 개발 인식”, “안전 개발 관련 지식이나 정보”, “안전 개발 프로세스” 순으로 조사되었다.

- 자동차

자동차 분야에서는 소프트웨어 안전을 위해 중요하다고 생각되는 항목에 대한 조사에서는 “안전 개발 인식”, “안전 개발 관련 지식이나 정보”, “안전 개발 프로세스” 순으로 조사되었다.

- 철도

철도 분야에서는 소프트웨어 안전을 위해 중요하다고 생각되는 항목에 대한 조사에서는

“안전 개발 관련 지식 및 정보”, “안전 개발 인식”, “코딩 표준” 순으로 조사되었다. 기타 의견으로 소프트웨어 개발 일정 확보 및 비용 확보라는 의견도 조사되었다.

- 항공

항공 분야에서는 소프트웨어 안전을 위해 중요하다고 생각되는 항목에 대한 조사에서는 “안전 개발 관련 지식이나 정보”, “개발 인식”, “안전 개발 프로세스” 순으로 조사되었다.

산업 분야별로 종합분석하면 법제도 개선, 정보 공유 시스템, 전담 기관 설립, 전문 인력 양성, 전문 기업 육성에 대한 필요성이 비표준 분야, 자동차 분야, 철도 분야, 항공 분야 모두 절대적으로 필요한 것으로 조사되었다.

SW안전 확보를 위해 다음 중 필요하다고 생각되는 항목은 무엇입니까?.

1. 법제도 개선
2. 기업 개발 프로세스 적용
3. 정보를 공유할 수 있는 시스템
4. 전담 기관 설립
5. 위험분석 전문인력 양성 (위험원 분석 및 평가 수행)
6. 안전관리 전문인력 양성 (안전 프로세스 관리)
7. 안전평가 전문인력 양성 (SW가 안전무결성수준에 부합하는지 평가)
8. 안전개발 전문인력 양성 (안전관련 활동 이해, 기능안전 지식·경험 보유)
9. 안전관련 전문기업 육성
10. 지속적인 홍보 및 교육
11. 기타( )

<표 4-159> 소프트웨어 안전 확보를 위해 필요한 항목 순위

산업분야	순위	소프트웨어 안전 확산에 필요한 항목
비표준	1순위	지속적인 홍보 및 교육
	2순위	안전개발 전문인력 양성
	3순위	정보를 공유할 수 있는 시스템
	4순위	위험분석 전문인력 양성
자동차	1순위	지속적인 홍보 및 교육

	2순위	안전관련 전문기업 육성
	3순위	안전개발 전문인력 양성
	4순위	안전관리 전문인력 양성
철도	1순위	법제도 개선
	2순위	지속적인 홍보 및 교육
	3순위	안전관련 전문기업 육성
	4순위	안전관리 전문인력 양성
항공	1순위	법제도 개선
	2순위	안전관련 전문기업 육성
	3순위	지속적인 홍보 및 교육
	4순위	안전관리 전문인력 양성

- 비표준 분야

비표준 분야에서는 소프트웨어 안전 확보를 위한 방안에 대한 우선순위 조사에서 “지속적인 홍보 및 교육 “안전 개발 전문 인력 양성” 순으로 조사되었다.

- 자동차

자동차 분야에서는 소프트웨어 안전 확보를 위한 방안에 대한 우선순위 조사에서 “지속적인 홍보 및 교육”, “안전 관련 전문기업 육성” 순으로 조사되었다.

- 철도

철도 분야에서는 소프트웨어 안전 확보를 위한 방안에 대한 우선순위 조사에서 “법제도 개선”, “지속적인 홍보 및 교육” 순으로 조사되었다.

- 항공

항공 분야에서는 소프트웨어 안전 확보를 위한 방안에 대한 우선순위 조사에서 “법제도 개선”, “안전관련 전문 기업 육성” 순으로 조사되었다.

산업 분야별로 종합분석하면 비표준 분야와 자동차 분야에서는 지속적인 홍보와 교육이 우선순위가 상대적으로 높았고, 철도와 항공에서는 법제도 개선의 우선순위가 상대적으로 높은 것으로 조사되었다.

제3절 종합 의견

소프트웨어 안전 일반 사항에 관한 공통 설문 부분을 종합적으로 분석하면 철도, 항공 부분은 안전에 대한 의식, 준비, 활동 등이 높은 수준을 보였으며, 자동차는 중간 수준, 비표준 분야는 조금 부족한 것으로 조사되었다. 비표준 분야는 소프트웨어 개발 프로세스에 대한 지식이 있고, 어느 정도 개발 프로세스에 따라 개발을 수행하는 것으로 조사되었으나, 위험 분석과 평가, 안전 관리에 대해서는 인식, 지식, 및 수행 측면에서 부족한 것으로 조사되었다.

세부적으로 분석하면 질문2에서 항공, 철도는 발주처가 국제표준이나 자체표준을 이용하여 기능안전을 제시하도록 요구하고 있는 것으로 조사되었다. 이에 따라 각 단계별 안전 활동이나 기법 적용 경험이 자동차나 비표준 분야에 비해 높은 수준으로 조사되었다. 질문4의 안전성 확보 이유에 대해서는 안전이 제품의 기본 기능이라는 대답이 가장 많아, 안전에 대한 중요성은 담당자들은 어느 정도 인식하고 있는 것으로 조사되었다.

질문 5의 세부 질문을 살펴보면 조사대상 모든 산업부분에서 소프트웨어 안전 관리자의 개발 프로세스 참여 형태는 독립적인 조직보다는 프로젝트에 속하여 하는 경우가 많아, 제3자 검증에 대한 필요성이 제기되었다. 또한 안전 개발 프로세스 적용 미비, 안전 계획 수립 미비, 산출물 관리 미비의 주요 원인이 전문 인력 부족으로 조사되어, 안전 관련 전문 인력 양성이 시급한 것으로 보인다.

질문 8의 효과적인 위험 분석 방법에 대해서는 안전성 분석 기법 사용, 사내 전문가 경험, 외부 컨설턴트 이용 등 항목에 대한 각 산업별 공통점을 찾기 어려워, 각 항목에 대한 세부 지식을 각 산업별로 공유하고 각 산업별 방법 적용에 대한 제한점 해결 방법을 구해야 한다. 예를 들면 비표준의 경우는 위험분석에 대한 지식에 부족함에도 외부 도움보다는 사내 전문가의 도움을 많이 받고 있는데, 이는 비용 및 기간 등의 문제로 외부 전문 컨설턴트 요청이 부진한 것으로 해석된다. 소프트웨어 위험 분석 시 사용하는 기법은 과거의 선형적인 위형 분석기법인 FTA, FMEA 등을 가장 많이 사용하는 것으로 조사되어, 소프트웨어의 복잡성으로 인한 위험 분석에 취약한 것으로 나타났다.

질문 9의 소프트웨어안전 무결성 등급은 발주처에서 제공하는 대로 적용하는 것으로 조사되어, 소프트웨어안전 무결성 등급은 시스템의 안전 무결성 등급을 상속하는 것으로 나타났다.

각 산업별 소프트웨어 프로세스 실태 조사 결과 6점 척도에 대한 결과는 공통 설문 결과와도 조금 다른 양상을 보인 항들이 보였다. 비표준 분야가 각 프로세스 단계 활동에서 높은 수준으로 대답한 부분이 보이는데, 이는 비표준 분야와 각 산업분야의 안전 요구에 대한 수준 차이로 이와 같은 결과가 나온 것으로 해석된다. 예를 들면 “요구사항 명세 단계의 산출물이 소프트웨어 설계에 필요한 정보를 제공하고 있나?” 하는 질문에서 자동차, 철도, 항공 부분은 “아니다”의 응답이 포함되어 있는 데 반해, 비표준 분야는 대부분 “그렇다” 라고 응답하였다. 그런데 비표준 분야는 요구사항 명세 산출물로 기본적인 요구사항만을 관리하고 있으며, 안전 관련 산출물의 관리하는 기업이 소수 인 것으로 위와 같이 분석이 가능하다.

철도, 항공 분야는 발주자의 요청에 따라 안전 요구 수준을 맞추어야 하며, 자동차 부분은 제품의 기본 기능으로 안전이 필수적이기 때문에 안전에 대한 요구 수준을 높을 것으로 예상된다. 이 조사에서 자동차 부분은 철도나 항공 부분에 대해 각 프로세스에 대한 수행도 및 안전 확보 활동이 미약한 것은 법제화가 철도나 항공 보다 미약하고, 조사대상이 소기업이 많은 부분 포함되어 있기 때문이기도 하다.

각 산업별 조사 결과, 공통적으로 나타나는 현상에 대해 정리해 보겠다. 요구사항 명세 단계에는 상위 시스템의 안전 요구사항과 인터페이스를 분석하고 만족시키는 활동이 가장 중요한 활동으로 조사되었으며, 요구사항 명세의 추적성 관리를 통해 이를 만족시켰다. 요구사항 명세를 위한 기법으로 자동차는 비정형 표기법, 철도와 항공은 준정형 표기법을 가장 많이 이용하였다. 각 안전 표준은 안전 등급에 따라 최소한으로 요구하는 기법 등을 요구하고 있으며, 자동차, 비표준 분야에도 준정형, 정형 기법을 도입하기 위한 교육이 필요할 것이다.

아키텍처 설계 활동에서 가장 중요한 원칙은 소프트웨어 아키텍처 명세를 작성 시 소프트웨어 요구사항 명세를 기초로 한다는 것이다. 그 다음 중요한 원칙은 하드웨어와 소프트웨어 상호 작용을 식별하고 분석하여 위험 요소를 제거하는 것이다. 안전 확보를 위해 수행된 활동은 소프트웨어 안전 무결성 등급의 요구사항이 구현 가능한지 분석하는 활동이다. 만약 요구사항 구현이 어려운 경우는 충동되는 요구사항을 변경하는 작업이 아키텍처 명세 시 수행되어야 한다.

상세 설계 단계에서도 요구사항 명세와 같이 검증을 위해서 정형 검증이나 준정형 검증보다는 설계에 의한 워크스루 방식을 많이 사용했다. 자동차나 항공 표준은 각 모듈을 모델화하고 검증을 자동화 하려는 시도가 일어나고 있다는 점과 비교하면 교육이 필요한 부분이 도출된다.

구현 단계에서는 소스 코드는 상위 문서를 기반으로 개발자의 책임 하에 구현하고 있었

다. 그러나 안전 확보를 위한 매카니즘 구현 활동은 부족했다. 시험 단계에서도 기본적으로 요구사항에 기초한 시험을 하고 있으며, 안전 확보를 위한 시험은 부족했다.

각 산업별 조사 결과, 비표준 영역과 표준 영역의 결과의 차이가 확연히 드러나는 부분이 있었다. 비표준 부분은 각 프로세스별 공통질문에서는 긍정적인 답을 했으나, 각 활동, 기법, 산출물에 대한 상세 질문 결과 요구사항 명세, 아키텍처 설계 부분은 제대로 시행하지 않는 것으로 확인되었다. 비표준 분야는 상세 설계 단계에서도 안전 무결성 검사나, 방어적 프로그램 사용 등의 안전을 위한 활동이 미약한 것으로 나타났다. 이러한 원인으로 소프트웨어 공학 지식의 부족에 대한 응답이 많았다.

철도 분야는 경우는 아키텍처 설계 단계에서부터 통합테스트 명세를 작성하고 준비하는 것으로 조사되었다. 각 분야 중 소프트웨어 프로세스를 가장 정확하게 적용하고 있는 것이다.

자동차 분야는 상세 설계 단계에서 변수 초기화 설계, 변수 이름을 다목적으로 사용하지 않음, 전역 변수와 포인터 제한 사용 등의 설계 원칙을 준수하고 있는 데, 이는 다른 분야에 확산 적용할 수 있다.

소프트웨어 안전 확산 방안 조사에서는 안전 프로세스 각 단계 활동이 잘 지켜지지 않은 이유가 각 분야별로 담당자의 안전에 대한 전문 지식 부족과 비용 및 시간의 부족이라고 조사되었다. 이는 각 단계 개별 질문의 결과와 동일하게 나타났다. 철도의 경우는 최신 기술을 사용하기 때문에 구현 방안이 없다는 응답이 많았는데, 이 또한 담당자의 안전에 대한 전문 지식의 일부분으로도 파악할 수 있다. 이러한 문제 해결을 위해서 참여인력의 안전 관련 의식 제고 및 안전 프로세스, 기법, 산출물 등 충분한 교육 제공의 필요성을 조사자들은 피력하였다.

각 분야별로 안전 확보를 위해 필요하다고 생각하는 것은 안전 확보를 위한 지식 및 정보라고 조사되었다. 이는 각 도메인의 특성에 맞는 안전 지식이 필요하고 이에 따라 소프트웨어 안전 개발 프로세스가 적용될 수 있다는 것이다.

본 프로세스 적용 실태 조사를 통해 그동안 추정하고 있던 사실들에 대한 좀 더 객관적인 데이터를 산출할 수 있게 되었다. 안전 인식에 대한 문제가 계속적으로 제기되었고 이번 조사를 통해 비표준 분야의 안전 인식 제고의 필요성이 도출되었다. 안전 프로세스 적용에서 가장 어려운 점 중의 하나로 안전 전문 인력 부족 문제가 도출되었다. 구체적인 문제점 도출과 해결 방안은 다음 장에 정리하겠다.

제5장 문제점 분석 및 소프트웨어 안전 프로세스 확산 방안

제1절 문제점 분석

표준 산업 분야는 소프트웨어 안전 표준에 따른 소프트웨어 안전 개발 프로세스에 대한 준수 여부와 준수를 위한 산업 내의 문제점 분석 및 개선 방안을 도출하여 국민 안전을 위한 정책방향을 제시하고, 소프트웨어 안전 산업 확산을 위해 소프트웨어 안전 프로세스 적용을 위해 무엇을 지원해야 할 것인지에 대해 도출하고자 한다. 비표준 산업 분야에서는 산업 표준이 없기 때문에 기존 소프트웨어 개발 프로세스를 활용 시 문제점이 어떤 것들이 있었는지 도출하고 소프트웨어 안전 개발 프로세스를 적시에 적용을 위해 지원해야 할 사항이 무엇인지 도출하고자 한다.

1. 표준 분야

1) 소프트웨어 안전성 보증을 위한 기법 적용이 어려움

표준 분야 조사 분석 결과 기법 적용을 위한 공통 질문에서 요구사항 명세 단계에서 표준 산업 평균 36%만이 안전성 보증을 위한 기법을 충분하게 적용하고 있고, 아키텍처 설계 단계에서도 40%만 안전성 보증을 위한 기법을 활용하고, 상세 설계 단계에서 안전성 보증 기법 적용은 36%로 것으로 조사되었다. 각 기법 적용에 대한 세부 질문 결과는 적용도가 더욱 낮은 것으로 조사되었다.

요구사항 명세 단계에서 안전성 보증을 위해 제어흐름도, 데이터 흐름도, 상태 천이도, UML등을 사용하고 있으나 정형 표기법은 전혀 사용하지 않고 있는 것으로 나타났고, 안전 요구사항 관리에 대한 자동화 툴 활용도 낮은 것으로 조사되었다. 또한, 자동화 활용이 낮아 자동화를 통한 지속적인 기법 적용과 안전성 보증을 체계적으로 확보할 필요가 있는 것으로 보인다.

아키텍처 설계 단계에서 안전성 보증을 위한 기법 적용은 자연어와 비정형 표기법을 많이 활용하고 있고, 정형 기법 등은 많이 활용되지 않은 것으로 나타났고, 자동화된 툴 활용은 저조한 것으로 조사되어 자동화된 툴을 활용 할 수 있는 기반을 마련할 필요가 있다.

상세 설계 단계에서 안전성 향상을 위한 검증 기법을 워크스루, 인스팩션, 정적 코드 분석을 통해 수행하고 있으나 각각 필요한 정형검증이나 의미적 코드 분석 등은 잘 수행하지 않은 것으로 조사되어 상세 설계 단계에서도 안전성 보증을 위한 충분한 기법을 적용할 필요가 있다.

2) 소프트웨어 안전성 확보를 상세 설계 과정에서 구현을 위한 충분한 설계가 이루어지지 않고 있음

상세 설계과정에서 소프트웨어 안전성 구현을 위한 충분한 설계가 이루지지 않는 것으로 조사되었다. 철도 분야는 72%로 높지만 자동차 분야와 항공 분야에서는 36%, 27%로 충분한 설계가 잘 이루어지지 않은 것으로 조사되었다.

소프트웨어 안전성 확보를 위한 충분한 설계가 잘 이루어지지 않고 있는 이유로는 제한된 시간과 비용에 따른 인력과 시간의 부족, 전문 개발자의 부족, 구현하고자 하는 코드가 복잡하여 설계단계에 많은 시간을 확보하기 어려움 등 자원의 한계에 따른 어려움이 있었다. 둘째는 설계, 공학적인 개념과 각 개발하고자하는 분야의 이해 부족, 소프트웨어 프로세스 각 단계별 세부적인 프로세스 지침서가 부족 등 지식의 한계에 오는 문제점이었다. 셋째는 개발자와 도메인 전문가의 소통의 문제였다. 마지막으로 설계 보다는 개발을 먼저 하는 관행과 같은 잘못된 인식이 원인이 되기도 했다.

3) 소프트웨어 안전 개발 프로세스 각 단계별 세분화된 프로세스 정보가 미흡

표준 분야에서 각 분야별 소프트웨어 안전성 표준에 따른 프로세스의 준수에 대한 절차상의 전체 평균은 47%로 낮은 것으로 조사되었고, 철도 분야는 65%, 자동차는 36%, 항공 분야는 41%로 소프트웨어 안전 개발 프로세스를 준수하는 것으로 조사되었다.

소프트웨어 안전 개발 프로세스에 대한 국제 표준 규격에 맞도록 정의된 상위 수준에서 소프트웨어 프로세스와 세부 활동, 산출물 등이 정의되어 있으나 실제 소프트웨어 안전 개발 시 적용할 수 있는 상세 단계의 시스템 안전성 평가 프로세스, 시스템 개발 프로세스, 안전성 도출 및 검증 기법 및 자동화된 도구 등에 대한 상세한 가이드나 내용이 정립되어 있지 않거나 정보가 오픈 되어 있지 않아 활용할 수 없다.

자동차 분야의 경우는 소프트웨어 안전 개발 프로세스는 요구사항의 정의, 아키텍처 설계,

소프트웨어 유닛 설계, 구현, 테스트에 대한 개발 프로세스를 통해 소프트웨어 안전 개발을 수행하고 있지만 세분화된 프로세스, 검증을 위한 기법, 산출물 정의를 위한 템플릿 등이 부족한 것으로 분석되었다.

철도 분야의 경우 소프트웨어 안전 개발 프로세스는 요구사항의 정의, 아키텍처 설계, 컴포넌트 설계, 구현, 테스트에 대한 개발 프로세스를 통해 소프트웨어 안전 개발을 수행하고 있지만 제한된 일정 및 예산으로 효과적으로 소프트웨어 안전 개발 프로세스 효과적으로 적용하기 위해서는 상세히 정의된 가이드 등이 필요하다.

항공 분야의 경우 소프트웨어 안전 개발 프로세스는 요구사항의 정의, 아키텍처 설계, 상세 설계, 구현, 테스트에 대한 개발 프로세스를 통해 소프트웨어 안전 기능을 구현하고 검증하고 있지만 소프트웨어 개발 각 단계별 세분화된 프로세스 정보가 미흡하여 요구사항이 충분히 적용되고 있지 못하는 경우가 있다.

4) 테스트 단계의 산출물이 계획 대비 소프트웨어 안전성 요구사항 추적이 어려움

표준 분야에서 테스트 단계의 산출물이 계획했던 소프트웨어 안전 요구사항을 완료했는지를 포함하는가에 대해 44% 정도만 잘 제시하고 있다고 조사되었다.

테스트 단계에서 소프트웨어 안전성 요구사항에 대한 충분한 반영이 되었는지에 대한 추적이 어려운 이유로는 자동차 분야에서는 초기에 요구사항인 하드웨어와 소프트웨어 인터페이스 명세서나 메모리에 대한 요구사항을 도출하고 지속적인 관리를 위해서는 적절한 일정 및 예산 투입이 되어야 하는데 현실적으로 일정과 예산 투입이 어려운 것으로 조사되었다. 철도 분야인 경우에도 일부 복잡한 시스템과 초기 시스템 개발 시 안전 요구사항의 정의가 불명확한 경우도 있기 때문에 요구 사항 도출이 어렵고 추적 관리도 잘 이루어지지 않는 것으로 조사되었다. 항공 분야인 경우 소프트웨어 할당 된 시스템 기능 및 인터페이스의 안전 요구 사항에 대한 분석과 지속적인 관리를 위한 상위 수준의 요구사항 표준을 준수하고, 변경 시 지속적인 변경 관리를 하도록 하고 있으나 시스템이 복잡하고 일정 및 예산적인 문제로 요구사항 추적을 지속적으로 관리하고 있지 않고 있다.

또한, 개발 요구사항과 안전 요구 사항이 이원화 되어 안전 요구사항이 개발에 정확하게 반영되고 있는지 관리가 어렵다. 일부 소프트웨어 안전 개발 프로세스에서 명확하게 정립되지 않았거나, 요건이 정확하게 도출되지 않아 요구사항 추적성을 확보하기 어려운 경우도 있는 것으로 조사되었다.

5) 소프트웨어 안전 개발 프로세스 적용에 따른 일정 및 예산 부족

표준 분야 조사 분석 결과 소프트웨어 안전 개발 프로세스에 대한 적용이 미비한 이유에 대한 가장 많은 답변이 소프트웨어 안전 개발 프로세스 적용을 위한 개발 일정과 개발을 위한 예산 부족으로 조사되었다. 소프트웨어 안전 개발 프로세스를 적용하여 국민의 안전을 확보하기 위해서는 안전에 적용되는 소프트웨어에 대해서는 충분한 개발 시간과 예산을 확보하여 국제 표준에 정하고 있는 프로세스를 지키고 개발을 수행할 필요가 있다.

소프트웨어 안전 개발 프로세스를 수행하기 위한 형상 관리나 변경 관리를 위한 시간과 예산도 충분히 확보하기 어렵고, 촉박한 일정으로 표준에서 정의하고 있는 충분한 테스트를 수행할 시간과 예산이 부족하다. 또한, 자체적으로 수행하기에는 예산 확보가 필요하기 때문에 고객이 요구하지 않거나 안전과 관련 없다고 판단되는 소프트웨어에 대해서는 관리를 잘 하지 않는 경우도 있는 것으로 조사되었다.

6) 소프트웨어 안전 개발 프로세스를 이해하는 전문가 부족

표준 분야 조사 분석 결과 소프트웨어 안전 개발 프로세스에 대한 적용이 어려운 이유는 전문가의 부족이다. 소프트웨어 안전 개발 프로세스를 적용하기 위해 투입되는 형태가 사내 독립적인 안전 관리 조직이 투입되는 경우도 있지만 많은 경우 개발업무와 겸임하고 있는 경우가 많은 것으로 조사되었다. 항공 분야의 DO-178C의 규격에 맞게 정의된 소프트웨어 세부 프로세스, 세부 활동, 산출물에 대한 충분한 이해와 활용을 할 수 있는 인력이 부족하고, 철도 분야에서도 한정된 예산과 일정으로 IEC 62279에 정의된 세부 개발 프로세스, 활동, 산출물을 잘 이해하고 활용할 수 있는 전문 인력이 부족하다. 자동차의 경우는 글로벌 제품인 경우에만 ISO 26262 국제 표준 규격에 정의된 소프트웨어 개발 프로세스, 세부 활동, 산출물 적용하기 때문에 관련 전문가가 절대적으로 부족하다.

2. 비표준 분야

1) 소프트웨어 안전 제품 개발 시 안전 계획 수립이 어려움

비표준 분야에서 소프트웨어 안전 제품 개발 시 안전계획을 수립하고 있는지 조사에서 20% 정도만 수행하는 것으로 조사되었다.

소프트웨어 안전을 위해서는 소프트웨어 안전 개발 프로세스 중 안전 계획 수립이 무엇보다 중요하다. 하지만 비표준 산업 분야에서는 소프트웨어 안전 계획에 대한 방법을 전혀 모르거나, 안전 계획을 수립하기 위한 전문 인력 부족, 안전 계획을 수립하기 위한 예산 부족이 대다수 인 것으로 분석되었다. 소프트웨어 안전 기능을 개발하기 위해서는 안전 계획을 수립하고 안전 기능을 적용하기 위한 요구사항을 도출해야 한다. 안전 계획 수립에 대한 인식 부족 및 필요성이 충분하기 않기 때문에 소프트웨어 안전성에 대한 정확한 개념이 부족하고 초기 소프트웨어 안전 개발 프로세스에 대한 안전 계획 수립이나 요구사항을 도출하는 과정을 잘 이해하지 못하는 것으로 조사되었다.

기존에 시스템 개발 시 발생하는 문제점인 시스템 Fail, 시스템 오작동, 제어부 구동 문제, 기준 정보 오류, 예외 처리, 보안 등 소프트웨어 안전 표준이 없기 때문에 발생하는 소프트웨어 안전 기능들을 정확하게 단계를 파악하고 해결 방안을 제시하기 위한 안전 계획 수립이 잘 이루어지지 않고 있다.

2) 소프트웨어 안전 개발 시 요구사항 도출 및 요구사항 추적이 어려움

비표준 분야 조사 분석 결과 소프트웨어 안전 개발 프로세스 단계 중 안전 관련 요구 사항에 대한 관리와 추적이 어려운 것으로 27% 정도가 잘 되고 있는 것으로 조사되었다.

비표준 분야에서는 요구사항 관리가 어려운 이유로 절대적 인력 부족, 비용 부족에 따른 인력 투입 어려움 등 인력에 대한 문제점을 지적했다. 두 번째로는 소프트웨어 공학에 대한 지식 부족과 지침 등의 미비로 적용이 어려운 점을 들었다. 또한 필요성에 대한 인식이 부족한 점도 있었다.

3) 소프트웨어 안전 개발 프로세스의 각 단계별 기법과 산출물 관리 미흡

비표준 분야에서 소프트웨어 안전성에 대한 공통 프로세스에 대한 설문에서 소프트웨어 안전 개발 프로세스 전체 평균은 27%로 조사되어 소프트웨어 안전성 개발 표준에 대한 각 단계별 프로세스와 기법에 대한 적용이 잘 안되는 것으로 조사되었다.

비표준 산업 분야에서 소프트웨어 안전 개발 프로세스 각 단계별 기법 적용 및 산출물에 대한 관리에 대해 미흡한 것으로 분석되었다. 산출물 관리가 미흡한 이유로 산출물 작성에 대한 방법을 모르거나 산출물 관리를 위한 일정, 비용, 지원이 부족한 것으로 조사되었다. 또한, 소프트웨어 위험 분석을 위해 사용되는 기법인 FTA, FMEA, HAZOP 등의 기법에 대한 이해가 부족하고 활용을 거의 하지 않는 것으로 조사되었다.

4) 소프트웨어 위험원 관리 및 위험성 평가 미흡

비표준 산업 조사 분석 결과로 소프트웨어 안전 프로세스 적용 시 소프트웨어 위험원 관리 및 위험성 평가에 대한 수준 조사 결과 이해 수준이 높은 경우가 7%로 표준 산업 분야보다 상대적으로 소프트웨어 위험원 관리 및 평가가 미흡한 것으로 나타났다. 소프트웨어 안전성을 확보하기 위해서는 위험원에 대한 도출과 평가가 필요하고 지속적인 관리가 필요하다.

5) 소프트웨어 안전 개발 프로세스 적용을 위한 환경 부족

비표준 분야 분석 결과 소프트웨어 안전 개발 프로세스는 프로젝트 인원에 속한 품질 담당자 등이 참여하고 소프트웨어 안전 개발을 준수하기 위한 프로젝트 참여 인원의 참여는 구분하기 어렵다.

그 이유는 고객의 강력한 요구나 표준 준수에 대한 필요성이 충분하지 않기 때문이라고 분석되었다. 소프트웨어 안전 개발 프로세스 적용을 필요성이 높지 않기 때문에 경험 있는 인력, 프로세스 적용 일정, 프로세스를 적용하기 위한 예산 등이 충분히 확보되지 않고 소프트웨어 안전 개발 프로세스를 수행하기 위해서는 충분한 환경이 필요하다.

6) 소프트웨어 안전 개발 프로세스 적용에 대한 경험 부족

비표준 산업 조사 분석 결과로 소프트웨어 안전 표준 공통인 IEC 61508을 기반으로 소프트웨어 안전 개발 프로세스를 일부 적용하고자 표준을 해석하고 적용을 수행하고자 하지만 공학적인 지식과 체계적인 개발 프로세스의 지식 및 경험에 대한 부족으로 적용에 많은 어려움을 겪고 있다. 일부 기업들은 소프트웨어 안전은 알고 있지만 소프트웨어 안전과 관련된 표준 등은 전혀 모르는 경우도 있다. 안전 표준과 적용 프로세스를 모르는 경우 소프트에어 안전 개발 프로세스 준용에 대한 제품 개발 요구 시 제품을 개발 할 수 없는 경우도 발생하고 있다.

7) 소프트웨어 안전 개발 프로세스를 이해하는 전문가 부족

비표준 분야 조사 분석 결과 소프트웨어 안전 개발 프로세스에 대한 적용이 안 되는 경우 중에 하나의 요인으로 프로세스를 잘 알고 있는 전문가의 절대적인 부족이다. 또한 제품별로 소프트웨어 안전 개발 프로세스 적용이 상이한데 이러한 상황별 대응을 위한 전문가를 찾기가 어렵기 때문에 소프트웨어 안전 개발 프로세스에 대한 적용이 잘 안되고 있다고 분석되었다.

내부 전문가의 부족에 따라 외부 전문가나 전문 업체에 위탁을 하길 원하지만 관련 분야의 전문가가 부족하기 때문에 적절한 지원을 받는 것이 어렵다는 의견이 다수로 조사되었다. 아직은 국제 표준 정립이 되어 있지 않지만 가까운 미래에 국제 표준이 제정될 것으로 예상하고 있기 때문에 소프트웨어 안전 관련한 전문가의 수급이 시급하다.

8) 소프트웨어 안전 개발을 위한 기본적인 공학 기술에 대한 이해 부족

비표준 분야 조사 분석 결과 소프트웨어 안전 개발 프로세스에 대한 인식이 낮기 소프트웨어 안전 개발에 대한 충분한 학습을 하지 않고 개발을 하는 경우가 많은 것으로 조사되었다. 또한, 소프트웨어 안전 개발을 위한 기본적인 공학 기술에 대한 이해도 낮기 때문에 기존 프로세스 적용 시 공학 기반의 프로세스를 잘 이행하기 않는 문제점도 나타났다. 기존 소프트웨어 개발 프로세스 기반을 두어 소프트웨어 안전 개발 프로세스를 적용하기 위해서는 기존 프로세스에 대한 공학적 접근이 필요하고, 소프트웨어 안전 개발 프로세스를 적용하고 확산을 위해서는 공학적인 인식과 기술의 습득도 필요한 것으로 분석되었다.

제2절 소프트웨어 안전 프로세스 확산 방안

1. 표준 분야

1) 소프트웨어 안전 개발 프로세스에 대한 지속적인 교육

국가 경쟁력 강화나 미래 사회에 중요한 요소로 작용할 소프트웨어에 대한 안전성을 유지하고 지키기 위해 소프트웨어 안전 개발에 대한 프로세스를 정확하게 숙지하고 개발할 수 있도록 할 필요가 있다.

안전이 중요한 소프트웨어를 개발 할 때에는 업무에 필요한 분석과 설계, 그리고 구현 및 테스트를 통해 소프트웨어 개발하고 고객이 지속적으로 사용하면서 개선 작업을 통해 소프트웨어의 완전성을 확보할 수 있다. 본 실태 조사결과 자동차 분야에서는 시스템 개발 시 위험 항목 관리에 대한 교육, 철도 분야에는 개발 관련 프로세스, 항공 분야에서는 안전 관리/인증 관련 프로세스에 대한 관리 및 교육이 필요하다.

표준 분야 공통으로 소프트웨어 안전 개발 프로세스 단계에서 안전성 보증을 위한 기법에 대한 교육이 필요하다. 정형기법, 자동화된 툴 적용 및 활용 방법, 안전 계획 및 안전 요구사항을 추출하고 테스트 단계에 활용할 수 있는 방법 등의 교육이 필요하다.

산업 분야별 표준을 잘 이해하고 정확하게 적용하여 소프트웨어 완전성을 확보하기 위해서는 각 산업별로 부족한 부분의 상세한 현황파악을 통한 지속적 교육과 다른 분야의 안전 확보를 위한 기술들에 대한 습득이 필요하다.

안전이 필요한 소프트웨어의 오류 시 심각한 인명 피해 및 사회적 문제가 발생할 수 있다는 사실은 인지하고 있으나, 시간적 제약, 환경적 제약 등으로 소프트웨어 안전 프로세스 적용에 어려움을 격고 있다. 이러한 문제점 해결을 위해서는 소프트웨어 안전 개발 프로세스의 필요성을 의사결정자부터 개발자까지 모두 인식할 수 있는 교육이 지속적으로 시행되어야 한다.

2) 소프트웨어 안전 개발 프로세스를 위한 전문 인력 확보

소프트웨어 안전 개발 프로세스 적용을 위해서는 소프트웨어 개발에 대한 경험을 풍부하

게 가지고 있어야 하고, 기본적으로 소프트웨어 공학에 대한 깊은 이해가 필요하다. 소프트웨어 안전에 대한 부분은 하드웨어와 같은 시스템에 속해 있기 때문에 하드웨어에 대한 이해도 필요하다.

산업 분야별로 소프트웨어 개발 프로세스에 대한 적용이 어려운 이유 중 하나로 전문 인력에 대한 부족이다. 산업 분야별로 제품(시스템)과 관련된 잠재적이고 비정상적인 동작을 신속하고 정확하게 식별하기 위해 내부 전문 인력을 활용하거나 외부 전문 인력의 도움을 받아 해결하는 경우가 많다. 하지만 산업군별로 소프트웨어 안전 관련 표준을 정확하게 인식하고 활용하는 전문 인력은 부족하다. 이러한 문제점을 해결하기 위해 적극적으로 소프트웨어 안전에 대한 관심을 갖고 전문가 양성을 위한 다양한 장치 마련과 예산 반영이 필요하다,

3) 소프트웨어 안전 개발 프로세스에 대한 가이드나 방법론 제공

소프트웨어 안전 개발 프로세스 적용을 위해서는 무엇보다 소프트웨어 안전 개발을 위한 개발 가이드가 필요하고, 활용할 수 있는 다양한 활용 예시, 산출물 템플릿을 제공할 필요가 있다. 표준 분야 기업들의 어려움 중에 하나는 소프트웨어 안전 표준을 이해하기 어렵다는 점이었다. 이러한 점을 해결하기 위해 개발자들이 쉽게 접하고 적용할 수 있는 가이드 개발에 대한 많은 투자가 필요하다.

소프트웨어 안전 표준에 대한 내용이 실무에 적용하기 어려운 점이 많기 때문에 표준을 해석하고 가이드를 제작할 때 실제 적용해서 사용하고 있는 사례를 적극적으로 발굴하여 추가한다면 표준 분야 기업에 많은 도움이 될 수 있을 것으로 판단된다.

산업분야별로 소프트웨어 안전 개발 프로세스 적용을 위한 일정, 비용, 인력, 자원이 절대적으로 부족하기 때문에 중소기업에서 활용 가능한 방법론과 가이드를 제작하여 배포할 필요가 있다.

4) 소프트웨어 안전 개발 프로세스 적용하기 위한 제도 장치 마련

소프트웨어 안전 개발 프로세스 적용을 위해서는 적용하는 기업들은 추가적인 인력, 비용, 일정의 증가가 반드시 필요하다. 영국 협회의 사례를 보면 IEC 61508 및 관련 표준을 입증하고 실무에 적용하는데 도움이 될 수 있도록 지침과 가이드를 개발하고 있다. 회원에는 최

종 사용자, 시스템 통합자, 하위 시스템 및 부품 제조업체, 인증 컨설턴트 및 인증기관이 포함되며, 영국의 보건안전처(HSE, Health and Safety Executive)도 참여하고 있다. 참여 회원들은 IEC 61508 표준에 맞는 업계 지침을 만드는데 우선적으로 참가하고, 규제 기관 및 평가 기관에 정책 및 의견을 제시한다. 협회 모임을 통해서 관련 전문가를 통해 지식 및 역량을 강화하고 멤버간 지식 공유 및 네트워킹으로 안전 표준 적용을 최적화 할 수 있도록 지원하고 있다. 또한, 영국 정부의 지원을 받아 이를 위한 도구들을 제공하기 위해 CASS(안전 관련 시스템 적합성 평가) 제도를 도입하여 인증 적합성 평가를 수행하고 있다. 이를 통해 인증 제공업체가 운영업체, 공급업체 및 규제기관이 신뢰할 수 있는 절차 및 기법을 제공하고 높은 수준의 안전 보증을 제공할 수 있게 되었다.

소프트웨어 안전성 확보를 위해서는 국내에서도 소프트웨어 안전 관련 표준을 관리하고 지속적으로 적용하기 위한 제도 장치가 필요하다.

2. 비표준 분야

1) 소프트웨어 안전 개발 프로세스에 적용을 위한 인식 개선 및 교육 수행

비표준 분야는 담당자들이 인식하는 것보다 안전한 소프트웨어를 개발하기 위한 준비가 되어 있지 않다. 우선 제품의 안전 확보에 소프트웨어 안전이 중요하다는 인식 개선이 선행되어야 한다. 제4차 산업혁명 이후 소프트웨어가 하드웨어를 제어하게 되면서 소프트웨어 안전에 대한 필요성이 강조되고 있다. 이러한 소프트웨어 안전 확보에 대한 인식 제고는 비표준 분야에서 가장 필요한 우선되어야 하는 교육이다.

다음으로 소프트웨어 안전 확보를 위해서는 소프트웨어 안전 개발 프로세스 적용이 필수적이라는 인식이 필요하며, 현재 시행하고 있는 소프트웨어 개발 프로세스보다는 좀 더 정교하고 체계적인 프로세스 적용이 필요하다.

비표준 분야에서는 소프트웨어 안전 개발 프로세스를 위해 형상관리나 변경 관리에 대한 인식이 낮고, 고객의 요구가 없는 경우 소프트웨어 안전 개발에 대한 프로세스를 적용하기 않기 않고 있다. 하지만 비표준 산업 분야도 가까운 미래에 소프트웨어 안전 개발 프로세스에 대한 적용을 요구할 것으로 예상하고 있기 때문에 철저한 준비가 필요하다.

2) 소프트웨어 안전 개발 프로세스를 수행 할 수 있는 인력 확보

소프트웨어 안전 개발 프로세스 적용을 위해서는 소프트웨어 개발에 대한 경험을 풍부하게 가지고 있어야 하고, 기본적으로 소프트웨어 공학에 대한 깊은 이해가 필요하다. 소프트웨어 안전에 대한 부분은 하드웨어와 같은 시스템에 속해 있기 때문에 하드웨어에 대한 이해도 필요하다. 소프트웨어 안전 개발을 위해 관련 국제 표준의 이해가 필요하고, 소프트웨어 품질에 대한 전문적인 지식도 필요하다.

비표준 산업 분야에서는 소프트웨어 안전 개발 프로세스 적용에 대한 충분한 필요성을 느끼지 못하기 때문에 상대적으로 수행 안전 개발 인력의 확보가 어렵다. 향후 필요한 소프트웨어 안전 개발 프로세스 전문 인력을 사전에 양성하고 투입 할 수 있다면 비표준 분야 산업에 크게 이마지 할 것으로 기대된다.

3) 소프트웨어 안전 개발 프로세스에 대한 표준 개발 및 가이드 제공

비표준 산업 분야는 각 분야에 대한 표준이 없기 때문에 어려움이 있다. 먼저 각 분야별 중요한 부분을 중심으로 표준을 제정하는 것이 선행되어야 한다. 그리고 표준을 바로 적용하는 것은 어려움이 있기 때문에 적용을 위한 기존 수행하고 있는 소프트웨어 안전 개발 프로세스를 정확하게 이해하고 공통 표준이나 타 산업 표준을 해석하고 적용 가능할 수 있는 가이드를 제작 할 필요가 있다. 또한, 간단하게 작성하고 활용 할 수 있는 사례를 적극적으로 발굴하여 제공한다면 중소기업에 많은 도움이 될 것으로 판단된다.

4) 소프트웨어 안전 개발 프로세스 적용하기 위한 법제도 장치 마련

비표준 산업 분야에서는 아직 소프트웨어 안전 개발 프로세스 적용에 대한 충분한 필요성이 있는 것은 아니다. 표준을 적용하기 위해서는 추가적인 인력, 개발 일정의 증가와 추가적인 비용이 들어가기 때문에 반드시 필요하다고 느끼지 않는다면 소프트웨어 안전 개발 프로세스를 적용하기 어렵다. 하지만 국민의 안전과 직접적으로 연관이 되는 안전 제품에 대해서는 강제적으로 지킬 수 있는 법제도적 장치도 필요하다.

하지만 강제적인 법제도 장치로 추가 비용이나 추가 일정 등으로 기업들의 어려움이 가중될 수 있기 때문에 소프트웨어 안전 개발 프로세스 적용을 위한 정부에서는 소프트웨어 안전을 관리 감독하고 지원 할 수 있는 전담기관도 고려하여 기업에 대한 부담을 최소하고 소프트웨어 안전은 최대화 할 수 있는 적극적인 지원과 투자가 필요할 것으로 판단된다.

제6장 결론

제1절 연구의 요약

소프트웨어 안전 개발 프로세스 적용 실태 조사를 조사하고 심층 분석을 위해 소프트웨어 안전 프로세스 관련 사례 조사 및 소프트웨어 안전 프로세스 조사를 위한 항목, 수준, 대상 선정 등 사전 작업을 수행하였다. 실태 조사를 위해 소프트웨어 안전 프로세스 적용을 하고 있는 영국의 CASS 사례를 조사하였고, 소프트웨어 품질 프로세스와 비교를 통해 소프트웨어 안전 프로세스 특징을 추출하였다.

또한, 심층 실태 조사를 위해 표준 산업군과 비표준 산업군의 실무 담당자가 이해하기 쉽게 하기 위해 쉬운 용어를 활용하여 조사지를 구성하여 50개 기업을 대상으로 설문과 대면 방문 조사를 통해 심층 인터뷰도 실시하였다.

조사지를 작성할 때 항공 분야는 DO-178C을, 자동차 분야는 ISO 26262, 철도 분야는 IEC 62279의 표준을 기반으로 조사지를 설계하였고, 비표준 분야는 IEC 61508을 토대로 조사지를 설계하였다. 설계된 조사지는 각 분야의 전문가의 검토를 통해 조사지를 확정하였다.

조사 대상은 표준 산업군으로 항공, 자동차, 철도 분야 있는 35개 업체를 대상으로 설문과 대면 조사를 통해 심층 조사를 실시하였다. 조사 결과 표준 산업군에 대한 소프트웨어 안전 프로세스에 대한 적용 정도는 표준 산업군에서는 77%이상 적용하고 있고, 비표준 산업군는 60% 정도가 적용한다고 조사되었다. 소프트웨어 안전 개발 프로세스에 대한 적용이 어려운 점으로는 전문 인력 부족하거나 수행하기 위한 예산 부족 등으로 나타났다.

소프트웨어 안전 개발에 대한 프로세스 적용을 확산하기 위해서 소프트웨어공학에 대한 기본적인 교육 제공, 안전성 관련 보증을 위한 비용 지원, 참여 인력의 안전 인식 제고, 법제도 지원 등이 필요한 것으로 나타났다.

제2절 시사점과 향 후 연구

본 실태조사를 통해 그동안 소수의 전문가를 통해 도출되었던, 소프트웨어 안전 인식 제고 필요성, 소프트웨어 안전 인력 부족 현상, 소프트웨어 안전 확보를 위한 추가 비용 문제 등에 대한 객관적 데이터를 가지게 되었다. 그러나 본 조사의 대상도 전체 소프트웨어 안전 기업과 인력을 대표하기 어렵기 때문에 국내 안전 소프트웨어 발전 방향을 도출하기 위한 정확한 데이터 도출이 필요하다. 향 후 연구에서는 소프트웨어 안전 활동을 모두 포함하는 소프트웨어 안전 산업을 정의하고 소프트웨어 안전 산업 내에 포함되는 기업, 인력 등의 수준을 파악하는 조사가 필요하리라고 본다.

제3절 연구의 한계

본 실태조사는 비표준 분야 15개 업체, 표준 분야 35개 업체에 대한 방문 조사를 통해 심층 조사를 수행하였다. 소프트웨어 안전 개발 프로세스에 대한 최초의 실태 조사이고 기업들의 현황을 사전에 파악하기 어렵기 때문에 대상 업체를 선정하고, 담당자를 만나서 심층 조사를 수행하는 것에 많은 시간과 노력이 투입되었다. 통계를 위해서는 비표준 분야와 표준 분야에 더 많은 자료가 필요하다. 그리고 소프트웨어 안전 개발 및 프로세스를 정립하기 위해서는 기존 소프트웨어 품질에 대한 비교 할 수 있는 조사도 필요할 것으로 판단된다.

부 록

I. 분야별 조사 설계 세부 항목

본 자료는 조사지 설계를 위해 각 표준(IEC 61508, DO-178, IEC 62279, ISO 26262)의 주요 항목을 정리한 것으로 실제 조사지는 본 항목 중 안전에 관련된 항목이 주로 포함되었다. 또한 본 조사의 목적에 맞도록 너무 상세한 부분은 제외하였다. 본 자료는 본 조사를 바탕으로 향 후 추가 조사를 위해 활용되기 위해 본 보고서에 첨부하였다.

1. 공통 분야 조사지 설계 세부 항목

1) 프로세스 조사 설계 세부 내용

<표> 조사지 공통 부문 조사항목

구분		유형	실태 조사항목(설문 문항)	추가 설명
표지	기업	기업 현황	기업명	
			주소 및 홈페이지	
			종업원수	전체직원수, 하드웨어 개발 인력수, 소프트웨어 개발 인력수, 연구소 인력수, 품질/안전 전담직원 수
			회사 업종	
			기업규모	중소기업,중견기업,소기업,기타
			주력산업분야 (1순위, 2순위, 3순위)	자동차,철도및지하철,항공,조선해양, 원자력,에너지,의료,기타( )
			매출액	전체 매출액, 소프트웨어 매출액
			주요 제품 분야 (1순위, 2순위, 3순위)	
			산업인증	국제인증, 민간인증, CMMI, ISO9000, GS인증, SP인증, 기타(), 없음
			소프트웨어비중	하드웨어와 소프트웨어대비
			수출유무	수출시 제품, 회사 비중
		작성 책임 자	이름	
			전화번호	
			소속부서 및 역할	
			이메일	
			직위	

			기업내 위치	경영자, 개발관리자, 하드웨어개발자, 소프트웨어개발자
			업무 경력	1~3년, 4~6년, 7~9년, 10년~12년, 13년~16년, 16년 이상
			나이	20대이하, 30대, 40대, 50대 이상
			자격증 보유	
안전일반	산업	안전	산업계에 기능안전에 대한 국제표준/법규제 존재 여부	DO-178C,ISO26262,IEC62279, 무기체계관리지침
			고객사에서 발주 시 기능안전 요구 여부	
	기업	일반	기업 - 소프트웨어 개발 방법론 보유 여부	
			기업 - 소프트웨어 수명주기 별 수준	
			개발하는제품이나시스템에서소프트 웨어 안전성확보가필요성	불필요, 불필요하나 확보되면 좋음, 중요하지만 사업에 영향이 적음, 매우 중요하고 사업에 영향, 반드시 필수요건
			개발하는 제품/시스템에서 소프트웨어 중요성(비중)은 어느정도 수준	중요하지 않음, 핵심 기능은 아니고 편의 기능, 핵심 기능의 일부, 핵심기능 전체, 기타( )
			소프트웨어 안전 관련 요건이 귀사의 제품 인허가나 수출에 필수적인 요건 여부	예, 아니오
		안전	기업 - 소프트웨어/하드웨어 안전이 필요하거나 구현된 제품이나 솔루션	
			기업 - 안전관리 담당자 보유 여부	
			기업 - 안전관리 프로세스 수립 여부	
			기업 - 위험 분석 산출물 관리 여부	
			기업 - 안전성 분석 기법 및 사용 예시	FMEA, FTA, HAZOP, STPA
			기업 - 안전성 분석 도구 보유 여부	
	제품	안전	소프트웨어제품 - 안전 무결성등급(SIL) 존재 및 예시	
			소프트웨어제품 - 안전 무결성등급(SIL) 산정 방법	
			하드웨어/소프트웨어 안전 기법에 대한 이해 전문가 보유 여부	

	지원	일반	형상관리 항목 변경 절차 및 통제활동 여부	
			소프트웨어 제품 개발 시 형상관리하는 산출물 목록	
			변경 요청 시 기록 관리되는 정보	
			사용중인 형상관리 도구	
			단계별 안전 관리 프로세스 및 책임	
			요구사항 관리소프트웨어 사용(명세, 추적)	
개선점		기반	기업 - 소프트웨어 안전이 중요하다고 생각하는가?	
			소프트웨어 안전 확보를 위해 중요하다고 생각하는 것은 무엇이라고 생각하십니까?	안전 개발 프로세스, 요구사항 명세, 테스팅, 코딩 표준, 위험도 분석, 기타()
			소프트웨어 안전 정보 취득 경로	세미나, 교육, 인터넷 검색, 전문가
			소프트웨어 안전 확보를 위한 개발 프로세스 적용이 필요여부	7점척도
		추진체계	소프트웨어 안전 확보를 위한 법제도 개선 필요 여부	7점척도
			소프트웨어 안전 확보를 위한 정보공유나 시스템 필요 여부	7점척도
			소프트웨어 안전 확보를 위해 전담 기관이 필요 여부	7점척도
		성장동력	소프트웨어 안전 확보를 위한 전문 인력에 대한 양성이 필요 여부	7점척도
			소프트웨어 안전 확보를 위한 위한 전문기업 육성이 필요 여부	7점척도

2. 자동차 분야 세부 항목

1) 프로세스 조사 설계 세부 내용

<표> 자동차 분야 안전개발 프로세스 조사 세부 항목

구분	유형	실태 조사항목(설문 문항)	선택 항목
소프	일반활동	요구사항	□시스템 요구사항에서 소프트웨어 요구사항으로

트웨어 요구사항 명세	안전활동	명세를 위해 수행하는 활동	할당여부 □요구사항 명세화 및 변경시 문서 반영 여부 □요구사항 명세 책임자 선임
		요구사항 명세 시 안전관련 수행 활동	□기술 안전 개념 □시스템 수준의 기술 안전 요구사항(TSR) 명세 □하드웨어-소프트웨어인터페이스명세서(HSI)(ECU커넥터,MCU/IC,메모리,주변장치) □소프트웨어 안전요구사항검증계획 □하드웨어-소프트웨어인터페이스검증계획
	기법	안전성 관점에서 사용하는 요구분석 기법	□자연어 □비정형표기법(그림,다이어그램,단순도표) □준정형표기법(제어흐름도,데이터흐름도,상태천이도,UML 등) □정형표기법(Zed,PVS,VDM등) □시스템안전요구사항과소프트웨어 안전요구사항간의추적성 □안전요구사항관리소프트웨어사용
	산출물	안전 요구사항 명세 관련 작성하는 산출물 목록	□안전계획 □모델링과프로그램밍언어를위한설계및코딩지침 □소프트웨어 안전요구사항명세서 □하드웨어-소프트웨어간인터페이스명세서(갱신) □소프트웨어검증계획(갱신) □소프트웨어검증보고서
소프트웨어 아키텍처 설계	일반활동	아키텍처 설계를 위해 수행하는 활동	□소프트웨어요구사항명세를기반으로소프트웨어아키텍처 명세를작성 □제안소프트웨어아키텍처를수립하고상세화 □모든하드웨어/소프트웨어상호작용을식별하고분석하여상세하게작성 □소프트웨어컴포넌트를식별 □소프트웨어와응용데이터/알고리즘간의상세한인터페이스를명시 □모델링과프로그래밍언어를위한설계및코딩지침
	안전활동	아키텍처 설계 시 수행하는 안전관련 활동	□안전목표 및 안전 개념의 타당성 확인 □안전개념및안전요구사항의검증 □안전목표나안전요구사항의위배를야기할수있는결함및고장을포함한조건과원인식별 □결함이나고장의검출을위한추가요구사항의식별 □검출된결함이나 고장에 대한 필수 반응(동작/수단)결정 □안전관련차량시험을포함하여안전목표나안전요구사항이 준수된다는것을검증하기위한추가요구사항의식별
	안전활동	아키텍처 설계 시 사용하는 방법	□소프트웨어 컴포넌트 유닛까지 식별되는 계층적 구조로 분할 식별 □소프트웨어컴포넌트의제한된크기

			□인터페이스의제한된크기 □소프트웨어 안전요구사항에 대한 소프트웨어 컴포넌트 할당 □컴포넌트할당된 요구사항에 대한 ASIL부합개발 □각소프트웨어컴포넌트내높은응집도(cohesion) □소프트웨어컴포넌트사이제한된결합도(coupling) □적합한 스케줄링 특성 □인터럽트의 제한된 사용 □소프트웨어 컴포넌트 정적 설계 □소프트웨어 컴포넌트 동적 설계
	안전활동	소프트웨어 정적 아키텍처 설계 방법	□소프트웨어컴포넌트리스트에정의된전체소프트웨어컴포넌트들이표기 □UML classdiagram의사용 □Middleware,device driver와 같은 하드웨어 관련 컴포넌트들이 같이 표기 □ASIL이 할당되는 소프트웨어 컴포넌트의 경우 ASIL이 표기 □기타( )
	안전활동	소프트웨어 동적 아키텍처 설계 방법	□소프트웨어 컴포넌트 상세 설계 □소프트웨어 Task Structure 정의 □Global Variable 정의 □기타( )
	기법	아키텍처 설계 시 사용하는 표기법	□자연어 □비정형표기법(그림,다이어그램,단순도표) □준정형표기법(제어흐름도,데이터흐름도,상태천이도,UML 등) □정형표기법(Zed,PVS,VDM등) □아키텍처 설계시 사용 소프트웨어( )
	기법	아키텍처 설계 검증 방법	□설계에 대한 워크스루(walk-through) □설계에 대한 인스펙션(inspection) □설계의 동적 부분에 대한 시뮬레이션 □시제품생성 □정형검증 □제어흐름분석 □데이터흐름분석
	산출물	소프트웨어 아키텍서 설계시 산출물	□소프트웨어아키텍처설계명세서 □안전계획(갱신) □소프트웨어 안전요구사항명세서(갱신) □안전분석보고서 □종속고장분석보고서 □소프트웨어검증보고서(갱신)
소프	일반활동	소프트웨어	□비안전관련요구사항설계

트웨어유닛설계		유닛 설계를 위해 수행하는 활동	□각 소프트웨어 유닛의 기능 상세 설계 □각소프트웨어유닛의인터페이스(function prototype등)설계 □각소프트웨어유닛의execution trigger조건설계 □각소프트웨어유닛이사용하는data및data의structure설계 □안전관련요구사항의각소프트웨어유닛의 ASIL 설계
	일반활동	유닛(단위) 설계 시 사용하는 표기법	□자연언어 □비정형표기법(그림,다이어그램,단순도표) □준정형표기법(제어흐름도,데이터흐름도,상태천이도,UML 등) □정형표기법(Zed,PVS,VDM등)
	안전활동	소프트웨어유닛설계의설계속성(가독성,단순성,강건성등)을달성하기설계원칙준수	□ C언어(MISRA-C coding rule 적용) 개발 □ 서브프로그램과 함수에서 하나 진입점과 하나 종료점 □ 동적 객체 또는 변수를 사용하지 않음. 사용한다면 동적 변수 생성시 온라인 시험 □ 변수 초기화 □ 변수 이름을 다목적으로 사용하지 않음 □ 전역 변수를 사용하지 않음, 만약 사용해야 한다면 그 사용에 대해 정당화 □ 포인터의 제한된 사용 □ 묵시적 형 변환 없음 □ 숨겨진 데이터 흐름이나 제어 흐름 없음 □ 무조건적 점프 없음 □ 재귀호출 없음 □그래픽모델링방식개발 □ 변수 초기화 □ 숨겨진 데이터 흐름이나 제어 흐름 없음 □ 재귀호출 없음 □어샘플러 프로그래밍
	기법	사용되는 소프트웨어 유닛(단위) 설계 검증 기법	□모델기반구현시검증 □수동코드개발시검증( ) □ 워크스루(Walk-through) □ 인스펙션(Inspection) □ 준정형 검증 □ 정형 검증 □ 제어 흐름 분석 □ 데이터 흐름 분석 □ 정적 코드 분석 □ 의미적 코드 분석(Semantic code analysis)
	산출물	단위설계수행 관련산출물	□소프트웨어유닛(단위)설계명세서 □소프트웨어검증명세서 □소프트웨어검증보고서(갱신)
소프트웨	일반활동	구현시일반적 인수행활동	□규정한 내용에 대한 소프트웨어 유닛직접구현 □모델기반 개발을 통한 자동코드생성

어구현			□사용하는 자동화툴
	안전활동	시험 케이스 완전성 평가를 위한 요구사항 구문 커버리즈 측정 방법	□구문 커버리지 □분기 커버리지 □MC/DC(변경된조건/결정커버리지) □활용되는 커버리지 소프트웨어 자동화도구( )
	안전활동	소프트웨어 유닛 구현에 대한 적합한 시험 환경	□MIL(model in the loop)시험 □SIL(software in the loop)시험 □PIL(prodessor in the loop)시험 □HIL(Hardware in the loop)시험 □기타( )
	기법	소프트웨어유닛(단위)시험방법	□요구사항기반시험 □인터페이스시험 □결함주입시험(예.변수의변수값손상,변이코드또는CPU레지스터의값손상 등에주입시험) □자원사용시험(단,에뮬레이터가자원사용시험을지원하는경우에만사용) □모델과코드간비교시험(back-to-backtest)(시뮬레이션할수있는모델이필요) □측정을위한사용되는코드나디버깅활용
	기법	소프트웨어 유닛 시험을 위한 시험 케이스 도출 방법	□요구사항분석 □등가등급생성및분석 □경계값분석 □오류추측(전문가판단에의한수집된데이터기준)
	산출물	소프트웨어 유닛 구현 시 산출물	□구현된소프트웨어유닛 □소프트웨어 검증계획(갱신) □소프트웨어 검증명세서 □소프트웨어 검증보고서(갱신)
소프트웨어시험	일반활동	소프트웨어 통합 및 시험	□소프트웨어 엘리먼트의 통합여부 □소프트웨어 아키텍처설계 기반 임베디드 시스템 구현 여부 검증 □설정 데이터통합 □보정 데이터통합
	안전활동	소프트웨어 안전기능을 확보를 위한 통합 및 시험	□단계별 기능안전요건시험여부 □설정 데이터검증 □보정 데이터검증
	안전활동	시험 케이스 완전성 평가를 위한 요구사항 구문 커버리즈	□함수(function)커버리지 □호출(call)커버리지 □활용되는 커버리지 소프트웨어 자동화도구( )

		측정 방법	
	안전활동	소프트웨어 통합에 대한 적합한 시험 환경	□MIL(model in the loop)시험 □SIL(software in the loop)시험 □PIL(prodessor in the loop)시험 □HIL(Hardware in the loop)시험 □기타( )
	기법	소프트웨어 안전 요구사항 검증 실시를 위한 환경 구현	□임베디드소프트웨어가소프트웨어 안전요구사항충족검증 □루프내의하드웨어검증 □전자제어단위네트워크환경검증 □차량직접검증
	기법	소프트웨어 통합 시험 방법	□요구사항기반시험 □인터페이스시험 □결함 주입시험 □자원 사용 시험 □모델과 코드간 비교 시험
	기법	소프트웨어 통합 시험을 위한 시험 케이스 도출 방법	□요구사항분석 □등가 등급 생성 및 분석 □경계값 분석 □오류 추측
	기법	하드웨어, 소프트웨어 통합 검토시 시 요구되는 기법	□ 기능 및 블랙 박스 테스트 □ 성능 테스트 □ 하드웨어 / 소프트웨어 통합 및 하드웨어 / 소프트웨어 통합 테스트 사양에 대한 시스템과 소프트웨어 설계 요구사항 간의 전향 적 추적 성
	산출물	시험 단계 수행관련 산출물 보유 및 사용 여부	□소프트웨어검증계획(갱신) □소프트웨어검증명세서(갱신) □임베디드소프트웨어 □소프트웨어검증보고서(갱신)

2) 조사지에 들어갈 기법 세부 내역

〈표〉 자동차 분야 안전개발 프로세스에서 사용되는 기법

ISO 26262 Part		ISO 26262 Methods Name	
소프트웨어 아키텍처 설계	소프트웨어 아키텍처 설계를 위한 표기법	1a	비정형 표기법
		1b	준정형 표기법
		1c	정형 표기법
	소프트웨어 아키텍처	1a	소프트웨어 컴포넌트의 계층적 구조

	설계 원칙	1b	소프트웨어 컴포넌트의 제한된 크기
		1c	인터페이스의 제한된 크기
		1d	각 소프트웨어 컴포넌트 내 높은 응집도(cohesion)
		1e	소프트웨어 컴포넌트 사이 제한된 결합도(coupling)
		1f	적합한 스케줄링 속성
		1g	인터럽트의 제한된 사용
	소프트웨어 아키텍처 수준의 오류 검출을 위한 메커니즘	1a	입력과 출력 데이터 범위 확인
		1b	유효성 확인(plausibility check)a
		1c	데이터 오류 검출b
		1d	외부 모니터링 기능c
		1e	제어 흐름 모니터링
		1f	다양한 소프트웨어 설계
	소프트웨어 아키텍처 수준의 오류 처리를 위한 메커니즘	1a	정적복구메커니즘a
		1b	적절한데그러데이션(gracefuldegradation)b
		1c	독립적병렬중복c
		1d	데이터 정정 코드
	소프트웨어 아키텍처 설계 검증방법	1a	설계에대한워크스루(walk-through)a
		1b	설계에대한인스펙션(inspection)a
		1c	설계의동적부분에대한시뮬레이션b
		1d	시제품생성b
		1e	정형 검증
		1f	제어흐름분석c
		1g	데이터흐름분석c
소프트웨어 단위 설계 및 구현	소프트웨어 유닛 설계를 위한 표기법	1a	자연 언어
		1b	비정형 표기법
		1c	준정형 표기법
		1d	정형 표기법
	소프트웨어 유닛 설계 및 구현을 위한 설계 원리	1a	서브프로그램과 함수에서 하나의 진입점과 하나의 종료점
		1b	동적 개체 또는 변수를 사용하지 않음, 혹은 그렇지 않으면 동적 변수 생성시 온라인 시험
		1c	변수 초기화
		1d	변수 이름을 다목적으로 사용하지 않음
		1e	전역변수를사용하지않음,만약사용해야한다면그사용에대해 정당화한다.
		1f	포인터의 제한된 사용
		1g	암시적 형 변환 없음
		1h	숨겨진 데이터 흐름이나 제어 흐름 없음
		1i	무조건적 점프 없음
		1j	재귀 없음
	소프트웨어 유닛 설계 및 구현 검증방법	1a	워크스루(Walk-through)
		1b	인스펙션(Inspection)
		1c	준정형 검증
		1d	정형 검증
		1e	제어 흐름 분석
		1f	데이터 흐름 분석

		1g	정적 코드 분석
		1h	의미적 코드 분석(Semantic code analysis)
소프트웨어 단위시험	소프트웨어 유닛 시험방법	1a	요구사항기반시험
		1b	인터페이스 시험
		1c	결함 주입 시험
		1d	자원 사용 시험
		1e	모델과 코드 간 비교 시험(back-to-back test), 해당되는 경우
	소프트웨어 유닛 시험을 위한 시험 케이스 도출방법	1a	요구사항 분석
		1b	동치 클래스(equivalence class)의 생성 및 분석
		1c	경계 값(boundary value)의 분석
		1d	오류 추측
	소프트웨어 유닛 수준의 구조적 커버리지 지표	1a	구문 커버리지
		1b	분기 커버리지
		1c	MC/DC (수정된 조건/결정 커버리지)
소프트웨어 통합 및 시험	소프트웨어 통합 시험방법	1a	요구사항 기반 시험
		1b	인터페이스 시험
		1c	결함 주입 시험
		1d	자원 사용 시험
		1e	해당되는 경우 모델과 코드 간 비교 시험(back-to-back test)
	소프트웨어 통합 시험을 위한 시험 케이스 도출방법	1a	요구사항 분석
		1b	동치클래스(equivalence class)의 생성 및 분석
		1c	경계값(boundary value)의분석
		1d	오류 추측
	소프트웨어 아키텍처 수준의 구조적 커버리지 지표	1a	함수(function) 커버리지
		1b	호출(call) 커버리지
소프트웨어 안전 요구사항의 검증	소프트웨어 안전 요구사항 검증 실시를 위한 시험 환경	1a	루프 내의 하드웨어
		1b	전자 제어 단위 네트워크 환경
		1d	차량

3. 철도 분야 세부 항목

1) 안전개발 프로세스 조사 설계 세부 내용

<표> 철도 분야 소프트웨어 안전 개발 프로세스 조사 세부 항목

구분	유형	실태 조사항목 (설문 문항)	선택항목

구분	유형	실태 조사항목 (설문 문항)	선택항목
1.1 요구 사항 명세	일반 활동	고객 요구사항을 정의하기 위해 수행하는 활동.	□기능성,강건성과 유지보수성, 안전성, 효율성, 사용편의성, 이식성 같은 품질특성을 고려 □ '시스템요구사항명세서', '시스템안전요구사항명세서', '시스템아키텍처기술서'를 준수하여 기술 □필요한 기능으로 요구사항을 완전하게 기술하고 소프트웨어 개발생명주기동안 소프트웨어 요구사항이 추적가능하도록 기술 □소프트웨어 개발생명주기에서 각단계의 책임자에게 이해가능하도록 상세하게 설명 □타시스템과의 인터페이스를 식별하고 기술 □관련된 운영모드 기술 □프로그램가능한 전자장치의 모든행위에 관련된 모드(특히고장행위) 기술 □하드웨어와 소프트웨어간의 제약사항 기술 □소프트웨어 자체적인 고장과 오류에 대해 검지하고 보고하는 기능과 하드웨어 상태확인 기능 기술 □비안전기능(일반기능)을 명확하게 식별 □위의 활동을 수행하지 않음.
	안전 활동	시스템 안전요구사항을 정의하기 위해 수행하는 활동.	□시스템 안전 무결성 등급을 준수하기 위한 기능을 명확하게 식별 □위의 활동을 수행하지 않음.
	안전 활동	시스템 안전요구사항과 관련된 사항을 테스트하기 위한 활동.	□시스템 안전요구사항과 관련된 사항을 주기적으로 테스트하는 기능 기술 □시스템 안전요구사항과 관련해서 시스템운영중에 안전기능을 테스트할 수 있도록 기술 □위의 활동을 수행하지 않음.
	기법	다음의 기법 및 대책 중 요구사항단계에서 적용하는 것.(7.2.4.15)	□정형 기법(Formal Method) □정형 명세(Formal Specification) □정형 검증(Formal Verification) □모델링(Modeling) □데이터 모델링 □구조적 방법론(Structured Methodology) □결정테이블(Decision Tables) □위의 기법 및 대책을 적용하지 않음.
1.2 요구	일반 활동	소프트웨어 요구사항	□소프트웨어 요구사항적합성은 소프트웨어 품질보증계획에 따라 작성자와 별도의 조직에서 검토

구분	유형	실태 조사항목 (설문 문항)	선택항목
사항 검토		적합성 검토를 위해 수행하는 활동.	□요구사항 추적표를 작성하여 소프트웨어 요구사항이 시스템 요구사항 으로부터 도출되었는지 추적성 검토 □소프트웨어 요구사항 명세서 체크리스트를 통해 정확성/완전성, 일관성, 구현가능성, 표준준수성, 테스트용이성 검토 □위의 활동을 수행하지 않음.
1.3 종합 소프트웨어 테스트명세	일반 활동	종합 소프트웨어 테스트를 명세하기 위해 수행하는 활동	□개발 완료된 소프트웨어에서 수행할 테스트에 대해 목적, 테스트케이스, 환경, 완료기준 등 기술 □테스트케이스는 실행순서와 입력값, 출력값, 완료기준 등을 포함하여 기술 □위의 활동을 수행하지 않음.
	기법	소프트웨어 테스트를 위해 적용하는 대책 및 기법	□성능테스트(Avalanche/스트레스테스트, 반응시간 및 메모리제약사항, 성능요구사항) □기능테스트(경계값분석, 동치클래스 및 입력 파티션 테스트)의 조합 □위의 활동을 수행하지 않음.
1.4 요구사항 검증	일반 활동	요구사항 검증을 위해 수행하는 활동.	□소프트웨어 요구사항 검증 보고서 작성 □하드웨어와 소프트웨어 간의 제약사항을 반드시 고려하고, ‘소프트웨어요구사항명세서’ 와 ‘시스템요구사항명세서’, ‘시스템안전요구사항명세서’, ‘소프트웨어품질보증계획서’ 의 적합성 검증 □ ‘소프트웨어 요구사항 명세서’ 의 요구사항이 가독성과 추적성을 고려하여 기술했는지 검증 □테스트할 수 없는 요구사항의 정확한 커버리지를 증명하기 위한 추가 활동 정의 □소프트웨어 요구사항이 개발하고자 하는 시스템의 모습을 정확히 반영하고 있는지 검증 □ ‘소프트웨어 요구사항 명세서’ 에 대한 검증결과 및 부적합사항에 대한 해결책과 권고사항 명시 □위의 활동을 수행하지 않음.
	안전 활동	안전성 관점에서 요구사항 검증을 위해 수행하는 활동.	□안전성 관점에서 하드웨어와 소프트웨어간의 제약사항을 반드시 고려하고, ‘소프트웨어요구사항명세서’ 와 ‘시스템요구사항명세서’, ‘시스템안전요구사항명세서’, ‘소프트웨어품질보증계획서’ 의 적합성 검증 □위의 활동을 수행하지 않음.
2.1	일반	소프트웨어	□소프트웨어 요구사항명세를 기반으로 소프트웨어 아키텍처

구분	유형	실태 조사항목 (설문 문항)	선택항목
아키텍처 명세	활동	아키텍처 명세를 작성하기 위해 수행하는 활동.	명세 작성 □제안 소프트웨어 아키텍처를 수립하고 상세화 □모든 하드웨어/소프트웨어 상호작용을 식별하고 분석하여 상세하게 작성 □모든 소프트웨어 컴포넌트를 식별하고 각 컴포넌트에 대하여 다음을 확인한다: 신규/기존재 여부, 기존재 컴포넌트의 검증 여부 □소프트웨어 요구사항의 하위집합을 모두 포함하도록 소프트웨어 컴포넌트 식별 □소프트웨어 컴포넌트를 형상관리 시스템내에서 명확하게 식별하고 독립적으로 버전관리 □가능하면 표준에 따라 개발하고 검증한 기존 소프트웨어 컴포넌트 재사용 □소프트웨어가 응용프로그램데이터 또는 알고리즘으로 구성된 경우, 소프트웨어와 응용데이터/알고리즘간의 상세한 인터페이스 명시 □위의 활동을 수행하지 않음.
	일반 활동	소프트웨어 인터페이스 명세를 작성하기 위해 수행하는 활동.	□소프트웨어 인터페이스 명세에 ‘소프트웨어요구사항명세서’와 ‘소프트웨어아키텍처명세서’를 기반으로 소프트웨어 컴포넌트와 전체 소프트웨어의 경계에 대한 모든 인터페이스 기술 □인터페이스명세에 다음의 사항을 포함한다: 사전/사후조건, 특정데이터에 대한 경계값, 경계값 및 초과시 동작, 기능간 동기화 메커니즘 등 □위의 활동을 수행하지 않음.
	일반 활동	소프트웨어 개발 전략 수립시 수행하는 활동.	□요구 소프트웨어 안전 무결성등급에 따른 소프트웨어 개발전략 수립 □소프트웨어 개발전략은 다음과 같이 정제하고 구조화하여 소프트웨어 아키텍처명세에 수립한다: 주요 품질속성, 검증/시험/실행/유지보수가능성, 요구사항 명세 추적성 □위의 활동을 수행하지 않음.
	안전 활동	소프트웨어 아키텍처 명세 작성 시, 안전을 고려하여 수행하는 활동.	□요구 소프트웨어 안전 무결성 등급의 요구사항이 구현 가능한지 분석 □소프트웨어 아키텍처는 애플리케이션의 안전부분의 크기와 복잡성을 최소화하는 것을 목표로 함 □식별된 컴포넌트의 안전 무결성 등급을 확인 □기존 소프트웨어를 재사용하기 위해서 모든 소프트웨어 안전 무결성 등급에 대하여 다음 정보를 명확하게 식별하고

구분	유형	실태 조사항목 (설문 문항)	선택항목
			문서화한다: 관련요구사항, 환경에 대한 가정, 다른 소프트웨어와의 인터페이스 □모든 소프트웨어 안전 무결성 등급에 대해 전체 소프트웨어 검증 프로세스에 기존 소프트웨어를 포함하여 수행 □장애회피와 장애대응 전략 간의 균형을 이루어 장애처리수단을 수립 □선정한 기법, 대책 및 도구가 요구 소프트웨어 안전 무결성 등급의 소프트웨어 요구사항명세를 만족함을 증명 □위의 활동을 수행하지 않음.
	안전 활동	소프트웨어 안전 무결성 등급이 SIL 3 또는 SIL 4의 경우 수행하는 안전활동.	□기존 소프트웨어의 예상 가능한 실패 및 전체 소프트웨어에 대한 영향 분석 □기존 소프트웨어의 고장탐지 및 이러한 고장으로부터 시스템을 보호하기 위한 전략 정의 □기존 소프트웨어가 할당된 요구사항을 충족함을 검증 및 확인 □기존 소프트웨어의 오류가 감지되고 기존 소프트웨어가 통합된시스템이 이러한 오류로부터 보호됨을 검증 및 확인 □기존 소프트웨어의 환경에 대한 가정이 성취 되었음을 검증 및확인 □위의 활동을 수행하지 않음.
	기법	소프트웨어 아키텍처 명세를 위해 적용하는 대책 및 기법.	□방어적 프로그래밍 □결함검출 & 진단 □오류정정코드 □오류검출코드 □고장단정 프로그래밍 □안전성백기법 □다양화 프로그래밍 □복구블록 □역방향복구 □전방향복구 □고장복구 재시도 방법 □실행된 사례기억 □인공지능-결함정정 □소프트웨어의 동적 재구성 □소프트웨어 오류 영향분석 □우아한 저하 □정보은닉

구분	유형	실태 조사항목 (설문 문항)	선택항목
			□정보캡슐화 □완전하게 정의된 인터페이스 □정형기법 □모델링 □구조적 방법론 □컴퓨터지원 설계 및 명세도구를 통한 모델링 □위의 활동을 수행하지 않음.
2.2 소프트웨어설계 명세	일반 활동	소프트웨어 설계 명세를 작성하기 위해 수행하는 활동.	□ ‘소프트웨어요구사항명세서’, ‘소프트웨어아키텍처명세서’ 및 ‘소프트웨어인터페이스명세서’를 기반으로 소프트웨어 설계명세를 작성 □소프트웨어 설계명세에 다음의 사항을 포함한다: 소프트웨어 컴포넌트와 외부환경과의 인터페이스, 소프트웨어 컴포넌트간의 인터페이스, 데이터구조, 컴포넌트에 대한 요구사항할당 및 추적, 주요 알고리즘과 처리순서, 오류보고 메커니즘 □코딩표준을 개발하고 명시 □모든 소프트웨어의 개발에 코딩표준을 사용하고, 소프트웨어 품질보증계획에서 참조 □위의 활동을 수행하지 않음.
	안전 활동	소프트웨어 설계 명세 작성시, 안전을 고려하여 수행하는 활동.	□설계 명세서 작성시, 소프트웨어 아키텍처와 그 안전 무결성등급에 대한 소프트웨어 컴포넌트의 추적성 기술 □코딩표준의 선택에 대하여 소프트웨어 안전 무결성등급 요구범위에서 충분한 근거 제시 □위의 활동을 수행하지 않음.
	기법	소프트웨어 설계를 위해 적용하는 대책 및 기법.	□상태전이 다이어그램 □시퀀스 다이어그램 □구조적 방법론 □분석가능한 프로그램 □엄격한 형식의 프로그래밍언어 □구조적 프로그래밍 □절차적 프로그래밍 □위의 활동을 수행하지 않음.
2.3 소프트웨어통합	일반 활동	소프트웨어 통합 테스트 명세를 작성하기 위해 수행하는 활동.	□소프트웨어 요구사항 명세, 소프트웨어 아키텍처 명세, 소프트웨어 설계 명세 및 소프트웨어 인터페이스 명세에 기초하여, 소프트웨어 통합테스트 명세 작성 □소프트웨어 통합테스트 명세 작성 시 다음의 사항 포함: 테스트 목적, 테스트케이스, 테스트유형, 환경, 완료판정기준,

구분	유형	실태 조사항목 (설문 문항)	선택항목
테스트명세			커버리지기준, 책임 □위의 활동을 수행하지 않음.
	기법	소프트웨어 통합 테스트 명세를 위해 적용하는 대책 및 기법.	□동적 분석 및 테스트 □기능/블랙박스 테스트 □성능 테스트 □위의 활동을 수행하지 않음.
2.4 소프트웨어/하드웨어통합테스트명세	일반 활동	소프트웨어 / 하드웨어 통합 테스트 명세를 작성하기 위해 수행하는 활동.	□시스템설계, 소프트웨어 요구사항 명세, 소프트웨어 아키텍처 명세 및 소프트웨어 설계명세에 기초하여 '소프트웨어/하드웨어통합테스트명세서' 작성 □통합테스트의 적절한 방향설정과 특정설계 또는 기타 통합요구가 적절하게 제공될 수 있도록 개발생명주기 초기에 '소프트웨어/하드웨어통합테스트명세서' 생성 □소프트웨어/하드웨어 통합 테스트 명세는 다음의 사항을 보장한다: 하드웨어에서 지정된 하드웨어 인터페이스를 통해 소프트웨어가 적절한 방법으로 실행됨, 소프트웨어는 하드웨어오류를 요구에 맞도록 처리할수있음, 요구타이밍과 성능에 대한 시연, 테스트케이스 작성, 컴포넌트테스트와 소프트웨어 통합 테스트결과를 소프트웨어/하드웨어통합테스트에 재사용할 경우 이를 명시 □소프트웨어/하드웨어 통합 테스트명세에 다음의 사항 포함: 테스트케이스와 테스트데이터, 수행할 테스트유형, 도구, 지원소프트웨어, 설정을 포함한 테스트환경, 테스트 완료 판정기준 □위의 활동을 수행하지 않음.
	기법	소프트웨어 / 하드웨어 통합 테스트 명세를 위해 적용하는 대책 및 기법.	□정형 증명 □정적 분석 □동적 분석 및 시험 □측정 기준 □추적성 □소프트웨어 오류 영향분석 □코드시험 적용범위 □기능적/블랙박스시험 □성능시험 □인터페이스시험 □위의 활동을 수행하지 않음.
2.5	일반	소프트웨어	□소프트웨어 요구사항명세, 소프트웨어 아키텍처명세,

구분	유형	실태 조사항목 (설문 문항)	선택항목
아키텍처 검증	활동	아키텍처 및 설계 검증을 위해 수행하는 활동.	소프트웨어 설계 명세, 소프트웨어 통합 테스트 명세 및 소프트웨어/하드웨어 통합 테스트 명세에 기초하여 소프트웨어 아키텍처 및 설계검증 보고서 작성 □소프트웨어 아키텍처 및 설계검증 보고서 작성 □소프트웨어 아키텍처, 인터페이스, 설계명세는 다음의 사항보장 □소프트웨어 통합 및 소프트웨어/하드웨어 통합 테스트 명세는 다음의 사항 보장 □위의 활동을 수행하지 않음.
3.1 컴포넌트 설계	일반 활동	소프트웨어 컴포넌트 설계 명세를 위해 수행하는 활동.	□소프트웨어 컴포넌트 설계명세서 작성 □소프트웨어 컴포넌트 설계명세서는 가독성과 테스트 용이성을 고려하여 작성 □소프트웨어 컴포넌트설계시 크기와 복잡도에 대한 균형이 고려되어 컴포넌트가 구현될 때 반영될 수 있도록 기술 □위의 활동을 수행하지 않음.
	기법	소프트웨어 컴포넌트 설계 명세를 위해 적용하는 대책 및 기법.	□정형 명세(Formal Specification) □데이터흐름 다이어그램 □모듈방식 □컴포넌트 □정보은닉화 □정보캡슐화 □매개변수개수제한 □설계 및 코딩표준 □분석가능한 프로그램 □엄격한 형식의 프로그래밍언어 □구조적 프로그래밍 □프로그래밍언어 □언어하위집합 □객체지향언어 □절차적 프로그래밍 □메타프로그래밍 □위의 활동을 수행하지 않음.
	일반 활동	소프트웨어 컴포넌트 설계 적합성 검토를 위해 수행하는 활동.	□소프트웨어 컴포넌트 설계적합성은 소프트웨어 품질보증계획에 따라 작성자와 별도의 조직에서 검토 □추적표를 작성하여 소프트웨어 컴포넌트설계가 소프트웨어 설계명세에서 도출되었는지 추적성을 검토 □ ‘소프트웨어 컴포넌트 설계명세서 체크리스트’를 통해 정확성/완전성, 일관성, 구현가능성, 표준준수성 검토

구분	유형	실태 조사항목 (설문 문항)	선택항목
			□위의 활동을 수행하지 않음.
3.2 컴포넌트 테스트설계	일반 활동	소프트웨어 컴포넌트 테스트 명세를 위해 수행하는 활동.	□소프트웨어 컴포넌트 테스트 명세 작성 □테스트는 다음 목적을 준수 가능하도록 설계 - 컴포넌트가 의도된 기능을 수행한다는 것을 증명(블랙박스 테스팅) - 의도된 기능을 수행하기 위해 컴포넌트 내부의 상호작용을 검토한다.(블랙/화이트박스테스팅) - 컴포넌트의 모든 부분은 테스트가 된다는 것을 증명한다. □위의 활동을 수행하지 않음.
3.3 컴포넌트 설계 검증	일반 활동	소프트웨어 컴포넌트 설계 검증을 위해 수행하는 활동.	□소프트웨어 컴포넌트 설계 검증보고서 기술 □소프트웨어 컴포넌트 설계가 확정된후 소프트웨어 설계명세를 준수하고 있음 검증 □ ‘소프트웨어 컴포넌트 설계 명세서’는 가독성과 추적성을 가지는지 검증 □ ‘소프트웨어 컴포넌트 테스트 명세서’는 ‘소프트웨어 컴포넌트 설계명세서’의 모든 테스트케이스를 기술했는지 검증 □ ‘소프트웨어 컴포넌트 테스트 명세서’는 가독성과 추적성을 가지는지 검증 □ ‘소프트웨어 컴포넌트 설계 명세서’는 ‘소프트웨어 설계명세서’를 소프트웨어 컴포넌트로 세분화하는 과정에서 다음과 같은 사항을 고려했는지 검증: 요구사항의 구현가능성, 테스트가능성, 유지보수성 □위의 활동을 수행하지 않음.
4.1 컴포넌트 구현	일반 활동	소프트웨어 컴포넌트 구현을 위해 수행하는 활동.	□소스코드는 ‘소프트웨어 컴포넌트설계 명세서’를 기반으로 개발자(Implementer)의 책임 하에 구현 □소스코드의 크기(size)와 복잡도(complexity)는 균형을 이루도록 구현한다: SIL등급별로 권장하는 소스코드복잡도(Cyclomatic Complexity)기준 예시: SIL1<50, SIL2<20, SIL3/4<10 □소스코드는 읽기쉽고, 이해하기 쉬우며, 테스트 가능함 □소스코드는 테스트가 시작되기 전에 형상관리시스템에 등록하여 변경이력관리 □위의 활동을 수행하지 않음.
4.2 컴포넌트	일반 활동	소프트웨어 컴포넌트 테스트를 위해	□컴포넌트 테스트 수행환경은 실제 동작환경에서 수행되거나 최대한 근접한 환경하에서 수행 □ ‘소프트웨어 컴포넌트테스트보고서’는 기작성된

구분	유형	실태 조사항목 (설문 문항)	선택항목
테스트		수행하는 활동.	보고서와 소소코드를 기반으로 테스터의 책임 하에 작성 □ ‘소프트웨어 컴포넌트 테스트보고서’ 는 테스트보고서작성에 필요한 일반 요구사항을 준수하여 작성 □테스트 결과, 소프트웨어 컴포넌트설계명세에 기술된 요구사항을 만족하는지 여부와 같은 항목들 포함 □각 컴포넌트별로 테스트커버리지 지표가 제공되고 요구되는 커버리지수준 대비 달성률을 기술한다. 할당된 SIL에 해당하는 테스트 커버리지수준만족 □위의 활동을 수행하지 않음.
	일반 활동	소프트웨어 컴포넌트 테스트 결과 평가를 위해 수행하는 활동.	□ ‘소프트웨어 소스코드검증 보고서’ 는 ‘소프트웨어검증보고서’ 작성 에 필요한 일반 요구사항들을 준수하여 작성 □소스코드와 ‘소프트웨어컴포넌트테스트보고서’ 작성완료시 구현의 적절성, 가독성, 추적성 등 검증 □소프트웨어 소스코드구현과 ‘소프트웨어 컴포넌트 테스트보고서’ 작성이 완료된 이후 ‘소프트웨어 소스코드검증보고서’ 기술 □위의 활동을 수행하지 않음.
4.3 컴포넌트 구현 검증	일반 활동	컴포넌트 구현 및 테스팅 검증을 위해 수행하는 활동.	□ ‘소프트웨어 검증보고서’ 는 컴포넌트 구현 및 테스팅단계의 산출물을 대상으로 검증자(Verifier)의 책임하에 작성 □위의 활동을 수행하지 않음.
5.1 소프트웨어통합	일반 활동	소프트웨어 통합을 위해 수행하는 활동.	□소프트웨어 컴포넌트 통합은 시스템통합테스트, 시스템테스트를 수행하기 위해 컴포넌트인터페이스, 결합소프트웨어의 적절한 복합구성으로 개별 또는 기테스트된 컴포넌트를 포함하여 점진적으로 수행 □소프트웨어와 하드웨어를 통합하는 중에 수정 또는 변경이 발생하면 시스템영향분석을 실시하여 관련된 모든 컴포넌트를 식별하고 필요한 재검증 활동들 수행 □소프트웨어 통합테스트보고서는 소프트웨어 통합테스트명세서를 기반으로 통합자(Integrator)의 책임 하 에작성 □소프트웨어 통합보고서는 테스트보고서에 대한 일반 요구사항(6.1.4.5)들을 준수하여 작성 □소프트웨어 통합테스트보고서는 다음 항목들을 준수하여 작성: 테스트목표부합여부, 결함, 반복적테스트수행, 테스트환경

구분	유형	실태 조사항목 (설문 문항)	선택항목
			□소프트웨어 통합테스트 보고서는 기능 및 블랙박스테스팅(Functional and Black-boxTesting), 성능테스팅(Performance Testing) T&M을 올바르게 선택하여 사용하였는지 여부 증명 □위의 활동을 수행하지 않음.
	일반 활동	소프트웨어 통합을 위해 적용한 기능 및 블랙박스 테스팅, 성능 테스팅 대책 및 기법.	□경계값 분석(Boundary Value Analysis) □동등분할테스팅(Equivalence Classes and Input Partition Testing) □위의 활동을 수행하지 않음.
5.2 소프트웨어/하드웨어통합	일반 활동	소프트웨어/하드웨어 통합을 위해 수행하는 활동.	□소프트웨어/하드웨어 통합테스트보고서는 소프트웨어/하드웨어 통합테스트명세서를 기반으로 통합자(Integrator)의 책임 하에 작성 □소프트웨어/하드웨어 통합테스트보고서는 테스트보고서에 대한 일반요구사항들을 준수하여 작성 □소프트웨어/하드웨어 통합테스트보고서는 다음 항목들을 준수하여 작성한다: 소프트웨어/하드웨어 통합테스트보고서는 소프트웨어 통합테스트명세서에 기술된 테스트목표 및 기준에 부합하는지 여부를 테스트결과로 작성: 테스트목표부합여부, 결함, 반복적테스트수행, 테스트환경 □소프트웨어/하드웨어 통합테스트보고서는 기능 및 블랙박스테스팅(Functional and Black-boxTesting), 성능테스팅(Performance Testing)T&M을 올바르게 선택하여 사용하였는지 여부 증명 □위의 활동을 수행하지 않음.
	기법	소프트웨어/하드웨어 통합을 위해 적용한 기능 및 블랙박스 테스팅, 성능 테스팅 대책 및 기법	□경계값 분석(Boundary Value Analysis) □동등분할 테스팅(Equivalence Classes and Input Partition Testing) □위의 활동을 수행하지 않음.
5.3	일반	통합 검증을	□소프트웨어 통합검증보고서는 소프트웨어,

구분	유형	실태 조사항목 (설문 문항)	선택항목
통합 검증	활동	위해 수행하는 활동.	소프트웨어/하드웨어 통합테스트명세서관련 테스트보고서를 기반으로 검증자(Verifier)의 책임 하에 작성 □소프트웨어 통합검증보고서는 검증보고서(Verification Report)에 대한 일반 요구사항들을 준수하여 작성 □소프트웨어 통합테스트보고서와 소프트웨어/하드웨어 통합테스트보고서가 작성되면, 다음 항목들 검증: 적절성, 가독성, 추적성등 □위의 활동을 수행하지 않음.
5.4 종합 소프트웨어 테스트	일반 활동	종합 소프트웨어 테스트를 위해 수행하는 활동.	□종합 소프트웨어 테스트보고서는 종합 소프트웨어 테스트명세서를 기반으로 테스터(Tester)의 책임하에 작성한다. 종합 소프트웨어 테스트명세서 작성요구사항은 종합소프트웨어테스트보고서에 반영 □종합 소프트웨어 테스트보고서는 테스트보고서를 작성하기 위한 요구사항을 준수하여 작성 □확인자(Validator)는 재량에 따라 특정한 추가 테스트를 정의하고 테스트를 수행한다. .시스템의 실제 사용자의 필요를 반영한 복잡한 시나리오로 시스템에 부하를 주는 테스트도 같이 수행 □모든 테스트와 분석결과들은 종합 소프트웨어테스트보고서에 기록 □종합 소프트웨어테스트보고서는 검증자의 책임 하에 작성 □소프트웨어는 실하드웨어연결, 운영인터페이스로 실시스템연결, 입출력신호의 시뮬레이션으로 테스트. 시스템에 요구되는 일반운영모드 및 비정상조건들에 대해서도 테스트한다. 시뮬레이션 입출력데이터는 사용된 경우 실입출력데이터와 다르지 않음을 증명 □위의 활동을 수행하지 않음.
5.5 소프트웨어확인	일반 활동	소프트웨어 확인을 위해 수행하는 활동.	□소프트웨어 확인보고서는 소프트웨어 확인계획에 근거하여 확인자(Validator)의 책임 하에 작성 □소프트웨어 확인보고서는 확인보고서를 작성하기 위한 일반 요구사항을 준수하여 작성 □소프트웨어 확인보고서는 부속서A에 따라 선택된 각 T&M조합에 대해 검토한 내용 포함 □발견된 결함 및 표준, 요구사항, 계획, 제약사항에 대한 모든 불일치 사항들은 소프트웨어 확인보고서의 별도장절에 명확히 식별하며, 안전 무결성 레벨을 고려해 평가되어 소프트웨어와 함께 제공되는 배포(Release)노트에 포함 □위의 활동을 수행하지 않음.

2) 조사지에 들어갈 기법 및 산출물 세부 내역

아래 기법 및 산출물 내용은 정보통신산업진흥원에서 발간한 “소프트웨어 신뢰안전 철도 가이드”를 참조하여 작성된 것으로 우편의 참조에서 가리키는 위치는 가이드에서의 위치이다. IEC62279에 대해 상세히 기술되어 있으니 보다 상세한 내용을 알고 싶다면 해당 가이드를 참조하기를 추천한다.

<표> 철도 분야 안전개발 프로세스에서 사용되는 기법

활동 구분		기법 및 대책	참조
활동	소프트웨어 요구사항명세	1. 정형기법(수학적인 접근법에 기반)	부록.B-28
		2. 모델링	상세 참조
		3. 구조적 방법론	부록.B-52
		4. 결정 테이블	부록.B-13
	소프트웨어 아키텍처	1. 방어적 프로그래밍	부록.B-14
		2. 결함 검출 & 진단	부록.B-26
		3. 오류 정정 코드	부록.B-19
		4. 오류 검출 코드	부록.B-19
		5. 고장 단정 프로그래밍	부록.B-24
		6. 안전성 백 기법	부록.B-47
		7. 다양화 프로그래밍	부록.B-16
		8. 복구 블록	부록.B-44
		9. 역방향 복구	부록.B-5
		10. 전방향 복구	부록.B-30
		11. 고장 복구 재시도 방법	부록.B-46
		12. 실행된 사례 기억	부록.B-36
		13. 인공지능-결함 정정	부록.B-1
		14. 소프트웨어의 동적 재구성	부록.B-17
		15. 소프트웨어 오류 영향 분석	부록.B-25
		16. 우아한 저하	부록.B-31

활동 구분		기법 및 대책	참조
		17. 정보 은닉	부록.B-33
		18. 정보 캡슐화	부록.B-33
		19. 완전하게 정의된 인터페이스	부록.B-38
		20. 정형기법	부록.B-28
		21. 모델링	상세 참조
		22. 구조적 방법론	부록.B-52
		23. 컴퓨터 지원 설계 및 명세도구를 통한 모델링	상세 참조
	소프트웨어 설계및구현	1. 정형 기법	부록.B-28
		2. 모델링	표 A.17
		3. 구조적 방법론	부록.B-52
		4. 모듈 방식	부록.B-38
		5. 컴포넌트	상세 참조
		6. 설계 및 코딩 표준	상세 참조
		7. 분석 가능한 프로그램	부록.B-2
		8. 엄격한 형식의 프로그래밍 언어	부록.B-49
		9. 구조적 프로그래밍	부록.B-53
		10. 프로그래밍 언어	상세 참조
		11. 언어 하위집합	부록.B-35
		12. 객체지향 프로그래밍	상세 참조, 부록.B-57
		13. 절차적 프로그래밍	부록.B-60
		14. 메타프로그래밍	부록.B-59
	검증 및 시험	1. 정형 증명	부록.B-29
		2. 정적 분석	상세 참조
		3. 동적 분석 및 시험	상세 참조
		4. 측정 기준	부록.B-37
		5. 추적성	부록.B-58

활동 구분		기법 및 대책	참조
		6. 소프트웨어 오류 영향 분석	부록.B-25
		7. 코드 시험 적용범위	상세 참조
		8. 기능적/블랙박스 시험	상세 참조
		9. 성능 시험	상세 참조
		10. 인터페이스 시험	부록.B-34
	통합	1. 기능 및 블랙박스 시험	상세 참조
		2. 성능 시험	상세 참조
	종합소프트웨어 시험	1. 성능 시험	상세 참조
		2. 기능 및 블랙박스 시험	상세 참조
		3. 모델링	상세 참조
상세	소프트웨어 분석	1. 정적 소프트웨어 분석	부록.B-13 부록.B-37
		2. 동적 소프트웨어 분석	
		3. 원인 결과 다이어그램	부록.B-6
		4. 이벤트 트리 분석	부록.B-22
		5. 소프트웨어 오류 영향 분석	부록.B-25
	소프트웨어 품질보증	1. ISO 9001 인증	
		2. ISO 9001 준수	
		3. ISO/IEC 90003 준수	
		4. 회사 품질 시스템	
		5. 소프트웨어 형상 관리	부록.B-48
		6. 체크리스트	부록.B-7
		7. 추적성	부록.B-58
		8. 데이터 기록 및 분석	부록.B-12
	소프트웨어 유지보수	1. 영향 분석	부록.B-32
		2. 데이터 기록 및 분석	부록.B-12
	데이터 준비	1. 테이블 명세 기법	부록.B-68

활동 구분		기법 및 대책	참조
		2. 응용 특화 언어	부록.B-69
		3. 시뮬레이션	부록.B-42
		4. 기능 테스트	부록.B-42
		5. 체크리스트	부록.B-7
		6. 페이건 정 검사	부록.B-23
		7. 정형 디자인 리뷰	부록.B-56
		8. 정형 증명 (데이타)	부록.B-29
		9. 워크스루	부록.B-56
	코딩 표준	1. 코딩 표준	부록.B-15
		2. 코딩 스타일 가이드	부록.B-15
		3. 동적 객체 사용 금지	부록.B-15
		4. 동적 변수 사용 금지	부록.B-15
		5. 포인터 사용 제한	부록.B-15
		6. 재귀호출 사용 제한	부록.B-15
		7. 조건 없는 점프 사용 금지	부록.B-15
		8. 함수, 서브루틴과 메소드의 크기와 복잡도 제한	부록.B-38
		9. 함수, 서브루틴에 대한 진입/종료 시점 전략 및 방법	부록.B-38
		10. 서브루틴 인자 수 제한	부록.B-38
		11. 전역 변수 사용 제한	부록.B-38
	동적분석 및 시험	1. 경계값 분석으로부터 테스트 케이스 수행	부록.B-4
		2. 오류 추축으로부터 테스트 케이스 수행	부록.B-20
		3. 오류 삽입으로부터 테스트 케이스 수행	부록.B-21
		4. 성능 모델링	부록.B-39
		5. 동등 클래스 및 입력 분할 테스팅	부록.B-18
		6. 구조 기반 시험	부록.B-50
	기능/블랙박스	1. 원인 결과 다이어그램으로부터 테스트 케이스	부록.B-6

활동 구분		기법 및 대책	참조
		수행	
	테스트	2. 프로토타입/애니메이션	부록.B-43
		3. 경계값 분석	부록.B-4
		4. 동등 클래스 및 입력 분할 테스팅	부록.B-18
		5. 프로세스 시뮬레이션	부록.B-42
	프로그래밍 언어	1. ADA	부록.B-54
		2. MODULA-2	부록.B-54
		3. PASCAL	부록.B-54
		4. C or C++	부록.B-54 부록.B-35
		5. PL/M	부록.B-54
		6. BASIC	부록.B-54
		7. Assembler	부록.B-54
		8. C#	부록.B-54 부록.B-35
		9. JAVA	부록.B-54 부록.B-35
		10. Statement List	부록.B-54
	어플리케이션 알고리즘	1. 기능적 블록 다이어그램	부록.B-63
		2. 순차적 함수 도표(차트)	부록.B-61
		3. 래더 다이어그램	부록.B-62
		4. 상태 차트	부록.B-64
	모델링	1. 데이터 모델링	부록.B-65
		2. 데이터 흐름 다이어그램	부록.B-11
		3. 제어 흐름 다이어그램	부록.B-66
		4. 유한 상태 기계 / 상태 전이 다이어그램	부록.B-27
		5. 시간 패트리넷	부록.B-55
		6. 결정/진리 테이블	부록.B-13

활동 구분		기법 및 대책	참조
		7. 정형 기법들	부록.B-28
		8. 성능 모델링	부록.B-39
		9. 프로토타입/애니메이션	부록.B-43
		10. 구조 다이어그램	부록.B-51
		11. 순차 다이어그램	부록.B-67
	성능시험	1. 과부하/스트레스 시험	부록.B-3
		2. 응답 시기 및 메모리 제약	부록.B-45
		3. 성능 요구사항	부록.B-40
	정적분석	1. 경계값 분석	부록.B-4
		2. 체크리스트	부록.B-7
		3. 제어 흐름 분석	부록.B-8
		4. 데이터 흐름 분석	부록.B-10
		5. 오류 추측	부록.B-20
		6. 워크스루/설계 검토	부록.B-56
	컴포넌트	1. 정보 은닉	부록.B-4
		2. 정보 캡슐화	부록.B-7
		3. 파라미터 수 제한	부록.B-8
		4. 모든 인터페이스 정의	부록.B-10
	코드테스트 커버리지	1. 구문	부록.B-50
		2. 분기	부록.B-50
		3. 복합 조건	부록.B-50
		4. 데이터 흐름	부록.B-50
		5. 경로	부록.B-50
	객체지향 소프트웨어 아키텍처	1. 아키텍처 클래스들에 대한 어플리케이션 도메인 컨셉의 추적성	-
		2. 일반적으로 사용되는 클래스들과 디자인패턴의 조합과 같은 적절한 프레임 사용	-

활동 구분		기법 및 대책	참조
		3. 객체지향 상세 설계	-
	객체지향 상세설계	1. 클래스들은 오직 하나의 목표만 가진다.	-
		2. 파생 클래스가 기본 클래스에 의해 구체화 되는 경우에만 상속이 사용된다.	-
		3. 상속의 깊이는 코딩 표준에 의해 제한된다.	-
		4. 메소드의 상속은 엄격한 제어 하에 수행	-
		5. 다중 상속은 오직 인터페이스 클래스들에만 사용된다.	-
		6. 알 수 없는 클래스로 부터의 상속	-

4. 항공 분야 세부 항목

1) 프로세스 조사 설계 세부 내용

(1) 요구정의

〈표〉 항공분야 안전개발 프로세스 조사 세부 항목 - 요구정의

설문 문항		선택항목	ref
1	상위 수준 요구사항을 정의하기 위해 수행하는 활동	(소프트웨어에 할당 된) 시스템 기능 및 인터페이스 요구 사항을 분석함.	5.1.2.a
		(소프트웨어에 할당 된) 시스템 요구 사항을 상위 수준 요구 사항으로 상세화함.	5.1.2.c
		시스템 위험 요소를 방지하기 위해 (소프트웨어에 할당 된) 시스템 요구 사항을 다루는 상위 요구 사항을 정의함.	5.1.2.d
		상위 수준 요구 사항은 소프트웨어 요구 사항 표준을 준수함.	5.1.2.e
		상위 수준의 요구 사항은 가능한 정량적으로 표현함. (적용할수있는허용오차를사용)	5.1.2.f
		상위 수준 요구 사항은 설계 또는 검증의 세부	5.1.2.g

		내용을 설명하지 않음.	
		인터페이스를 위한 매개변수(파라메타)항목을 사용하는 경우, 상위 수준 요구 사항에서 매개변수 데이터 항목이 소프트웨어에서 사용되는 방법을 기술 함.	5.1.2.j
		위의 활동을 수행하지 않음.	
2	상위 수준 요구사항과 시스템 요구사항 간의 추적성 관련하여 수행하는 활동	소프트웨어에 할당 된 시스템 요구 사항과 상위 수준 요구 사항 간의 양방향 연관을 보여주는 추적 데이터가 존재함.	5.5.a
		시스템 요구 사항이 모두 반영되었음을 확인함.	5.5.a
		시스템 요구사항과 상위 수준 요구사항의 양방향 추적성 확보 안됨	
3	요구사항프로세스에서 시스템프로세스와안전성평가프로세스와연관되어수행하는활동	적절하지 않거나 부정확하다고 판단 된 소프트웨어 요구 사항은 해당 요구사항이 발생된 프로세스(시스템 프로세스, 안전성 평가 프로세스)로 피드백 함.	5.1.2.b
		파생된 상위 수준의 요구 사항이 있는 경우, 해당 요구사항을 정의한 이유(reason)를 명시함.	5.1.2.h
		정의된 상위 수준 요구 사항은 시스템 안전성 평가 프로세스와 시스템 프로세스에 제공함.	5.1.2.i
		위의 활동을 수행하지 않음.	
4	상위 수준 요구사항 정의시 안전과 관련하여 정의하는 항목	안전 관련 요구 사항 및 잠재적 장애 조건	11.9
		고장(Failure) 탐지 및 안전 모니터링 요구 사항.	11.9
		소프트웨어에 할당 된 파티션 요구 사항, 파티션 된 소프트웨어 구성 요소가 서로 상호 작용하는 방법 및 각 파티션의 소프트웨어 수준	11.9
		상위 수준 요구사항 데이터가 없음.	
5	상위 수준 요구사항 데이터에 포함되어지는 내용	각 운영 모드에서의 기능 및 운영 요구 사항.	11.9
		성능 기준	11.9
		타이밍 요구 사항 및 제약 조건.	11.9
		메모리 사이즈 제약조건	11.9
		하드웨어와 소프트웨어 인터페이스	11.9
		상위 수준 요구사항 데이터가 없음.	
6	정의된 상위 수준 요구 사항을검증하기위해수행하는활동	각 상위 수준의 요구사항이 정확하고 모호하지 않으며 충분히 상세하고 서로 충돌되지 않음을 검증함.	6.3.1.b
		상위 수준 요구 사항과 대상 컴퓨터의 하드웨어 /소프트웨어 기능(특히 시스템 응답 시간), 입/출력 하드웨어간에 충돌이 없는지 검증함.	6.3.1.c
		정의된 상위 수준 요구 사항이 검증가능한지 검증함.	6.3.1.d
		소프트웨어 요구 사항 표준을 준수하고 표준에서	6.3.1.e

		벗어나는 것이 없음을 검증함.	
		제안된 알고리즘의 정확성과 동작을 검증함.	6.3.1.g
		상위 수준 요구사항 검증활동을 수행하지 않음.	
7	상위수준 요구사항과 시스템기능, 안전관련 요구사항이 반영되었는지를 검증	시스템 기능, 성능, 안전 관련요구사항이 상위 수준 요구사항으로 충분히 도출되었는지 검증함.	6.3.1.a
		시스템의 기능, 성능 및 안전 관련 요구 사항이 상위 요구 사항과 추적이 가능한지 검증함.	6.3.1.f
		상위 수준 요구사항이 시스템/안전 요구사항과의 추적성에 대한 검증을 수행하지 않음.	

⑵ 설계

〈표〉 항공분야 안전개발 프로세스 조사 세부 항목 - 설계

설문 문항		선택항목	ref
8	소프트웨어 아키텍쳐 정의시 수행하는 활동.	소프트웨어 아키텍처는 소프트웨어 설계 표준을 준수함.	5.2.2.a
		소프트웨어 구성 요소(component) 간의 인터페이스(데이터 흐름과 제어 흐름의 형태)를 일관성있게 정의함.	5.2.2.d
		소프트웨어 아키텍쳐를 정의하지 않음.	
9	정의된 소프트웨어 아키텍쳐를 검증하기위해수행하는 활동	소프트웨어 아키텍처가 상위 수준 요구 사항을 충족시키는지 검증함.	6.3.3.a
		소프트웨어 아키텍처의 구성 요소 간에 관계가 올바르다는 것을 검증함.	6.3.3.b
		소프트웨어 아키텍처와 대상 컴퓨터의 하드웨어 / 소프트웨어 간에 충돌이 없음을 검증함. (특히 초기화, 비동기 작업, 동기화, 인터럽트)	6.3.3.c
		소프트웨어 아키텍처가 검증가능함을 검증함.	6.3.3.d
		소프트웨어 설계 과정에서 소프트웨어 설계 표준을 준수하고 표준에서 벗어나는 것이 없을을 검증함.	6.3.3.e
		파티셔닝 위반이 없음을 검증함.	6.3.3.f
		소프트웨어 아키텍쳐 검증활동을 수행하지 않음.	
10	아키텍쳐 설계시 안전과 관련하여 고려하여 결정하는 내용	안전 관련 시스템 요구 사항을 추적 할 수있는 설계 결정에 대한 이론적 근거	11.10
		소프트웨어 로딩, 사용자 수정이 가능한 소프트웨어 또는 여러 버전의 다른 소프트웨어와 같은 구현 방법에 대한 설계 방법 및 세부 정보	11.10
		파티션 분할 방법 및 파티션 브랜치 방지 방법	11.10
		소프트웨어 구성 요소에 대한 설명 (새로운 구성인지 또는 이전에 개발되었는지, 이전에 개발	11.10

		된 경우, 취해진 기준을 참조).	
		아키텍쳐 설계시 안전을 고려하지 않음.	
11	상세요구사항 정의시 수행하는 활동	상세 요구 사항이 소프트웨어 설계 표준을 준수함.	5.2.2.a
		상세 요구 사항이 검증 가능하며 일관성을 유지하도록 함.	5.2.2.a
		수정 가능한 구성 요소를 변경하기 위해 제공된 수단은 수정 가능한 구성 요소를 변경할 수있는 유일한 수단으로 제시되어야합니다.	5.2.3.b
		비활성화 된 기능 또는 구성 요소가 활성화된 기능 또는 구성 요소에 부정적인 영향을 미치지 않도록 메커니즘을 설계하고 구현해야합니다.(deactivated fuction이 activated fuction에 영향을 주지 않도록 설계/구현되어야함.)	5.2.4.a
		사용을 의도하지 않은 환경에서는 비활성화 된 코드를 사용할 수 없다는 증거가 있어야합니다. 비정상적인 시스템 조건으로 인해 비활성화 된 코드가 의도하지 않게 실행되는 것은 의도하지 않은 활성 코드 실행과 동일합니다.	5.2.4.b
		활성 코드의 개발과 같이 비활성화 된 코드의 개발은 이 문서의 목표를 준수해야합니다.	5.2.4.c
12	소프트웨어 설계 프로세스 중 발견된 부적절한 요구사항에 대하여 유관 프로세스로 피드백하는 절차와 수행 방법	절차가 있으며, 절차에 따라 시스템 프로세스, 소프트웨어 요구사항 프로세스, 소프트웨어 계획 프로세스로 피드백 함.	5.2.2.g
		절차가 있으나, 시스템 프로세스, 소프트웨어 요구사항 프로세스, 소프트웨어 계획 프로세스로 피드백하지 않음.	
		절차는 없으나, 시스템 프로세스, 소프트웨어 요구사항 프로세스, 소프트웨어 계획 프로세스로 피드백 함.	
		절차도 없고, 시스템 프로세스, 소프트웨어 요구사항 프로세스, 소프트웨어 계획 프로세스로 피드백하지 않음.	
13	상세 요구사항과 상위 수준 요구사항 간의 추적성 관련하여 수행하는 행위	상위 수준 요구 사항과 상세 요구 사항 간의 양방향 연관을 보여주는 추적 데이터가 존재함.	5.5.b
		상위 수준 요구 사항이 모두 반영되었음을 확임함.	5.5.b
		상위 수준 요구사항과 상세 요구사항의 양방향 추적성 확보 안됨	
14	정의된 상세 요구 사항을검증하기위해수행하는활동	상세 요구 사항이 상위 수준 요구 사항을 충족시키고, 설계 기준을 올바르게 정의하였는지 검증함.	6.3.2.a
		상세 요구사항이 정확하고 모호하지 않으며 서로 충돌되지 않음을 검증함.	6.3.2.b

		상세 요구 사항과 대상 컴퓨터의 하드웨어 /소프트웨어간에 충돌이 없는지 검증함. (특히 버스 로드의 자원 사용, 시스템 응답시간, 하드웨어 입/출력)	6.3.2.c
		정의된 상위 수준 요구 사항이 검증가능한지 검증함.	6.3.2.d
		소프트웨어 설계 표준을 준수하고 표준에서 벗어나는 것이 없음을 검증함.	6.3.2.e
		상위 수준 요구 사항이 상세 요구 사항과 추적이 가능한지 검증함.	6.3.2.f
		제안된 알고리즘의 정확성과 동작을 검증함.	6.3.2.g
		상세 요구사항 검증활동을 수행하지 않음.	
15	안전성과 관련된 요구사항을 상세 요구사항으로 도출하기 위해 수행하는 활동	수정 불가능한 구성 요소(non-modifiable component)의 안전한 작동(safe operation)에 방해가되지 않도록 수정 가능 구성 요소(modifiable component)로부터보호될 수 있는 요구사항을 정의함.	5.2.3.a
		안전 관련 요구 사항은 제어 흐름 및 데이터 흐름을 주시하여 상세 요구사항을 정의 함.	5.2.2.e
		실패 조건에 대한 대응이 안전 관련 요구 사항과 일치하는지 확인함.	5.2.2.f
		상세 요구사항 도출시에는 안전관련 요구사항을 고려하지 않음.	
16	상세 요구사항 설계 시 시스템 프로세스와 안전성 평가 프로세스와 연관되어 수행하는 활동	소프트웨어 설계에서 파티션 또는 기타 아키텍처 수단을 사용하면 소프트웨어의 일부 구성 요소에 대한 소프트웨어 레벨 할당이 변경된 경우, 추가 데이터는 파생 된 요구 사항으로 정의함.	5.2.2.c
		파생된 상세 요구 사항이 있는 경우, 해당 요구사항을 정의한 이유(reason)를 명시함.	5.2.2.c
		파생된 상세 요구 사항은 안전성 평가 프로세스와 시스템 프로세스에 제공함.	5.2.2.c
		위의 활동을 수행하지 않음.	
17	설계문서(아키텍쳐 정의서 또는 상세 요구사항정의서) 에 포함되어지는 내용	상위 수준 요구 사항을 어떻게 충족시키는 지에 대한 자세한 설명	11.10
		소프트웨어 구조를 정의하는 소프트웨어 아키텍처에 대한 설명	11.10
		소프트웨어 아키텍처 전체에서 입/출력 설명	11.10
		설계의 데이터 흐름 및 제어 흐름.	11.10
		리소스 제한, 각 리소스 및 제한 사항을 관리하는 전략, 마진 및 마진을 측정하는 방법	11.10
		일정한 절차 및 프로세서 간 / 작업 간 통신 메커니즘	11.10

		소프트웨어 설계 프로세스에서 비롯된 파생 요구 사항	11.10
		시스템에 비활성화 된 코드가 포함되어있는 경우 대상 컴퓨터에서 코드를 활성화 할 수 없도록하는 방법에 대한 설명	11.10

(3) 코딩

〈표〉 항공분야 안전개발 프로세스 조사 세부 항목 - 코딩

설문 문항		선택항목	ref
18	소스 코딩(개발)시 수행하는 내용	소스 코드는 상세 요구 사항을 구현하고 소프트웨어 아키텍처를 준수함.	5.3.2.a
		소스 코드는 소프트웨어 코드 표준을 준수함.	5.3.2.b
		자동 코드 생성기의 사용시 소프트웨어 계획 프로세스에서 정의 된 제약 조건을 따라 사용 함.	5.3.2.d
19	코딩 프로세스 중 발견된 부적절한 요구사항에 대하여 유관 프로세스로 피드백하는 절차와 수행하는 방법	절차가 있으며, 절차에 따라 소프트웨어 요구 사항 프로세스, 소프트웨어 설계 프로세스, 소프트웨어 계획 프로세스로 피드백 함.	5.2.2.g
		절차가 있으나 소프트웨어 요구 사항 프로세스, 소프트웨어 설계 프로세스, 소프트웨어 계획 프로세스로 피드백하지 않음.	
		절차는 없으나, 소프트웨어 요구 사항 프로세스, 소프트웨어 설계 프로세스, 소프트웨어 계획 프로세스로 피드백 함.	
		절차도 없고, 소프트웨어 요구 사항 프로세스, 소프트웨어 설계 프로세스, 소프트웨어 계획 프로세스로 피드백하지 않음.	
20	소스 코드과 상세 요구사항 간의 추적성 관련하여 수행하는 행위	상세 요구 사항과 소스 코드 간의 양방향 연관을 보여주는 추적 데이터가 존재함.	5.5.c
		소스 코드가 문서화되지 않은 기능을 구현하지 않음을 확인함.	5.5.c
		소스코드가 모든 상세 요구 사항을 구현하였음을 확인함.	5.5.c
		상세 요구사항과 소스코드의 양방향 추적성 확보 안됨	
21	소스 코드를 검증하기위해수행하는활동	상세 요구 사항에 대해 소스 코드가 정확하고 완전하게 구현하였으며, 문서화되지 않은 기능을 구현하지 않았는지 검증함.	6.3.4.a
		소스 코드가 소프트웨어 아키텍처에 정의 된 데이터	6.3.4.b

		흐름 및 제어 흐름과 일치하는지 검증함.	
		소스 코드가 검증 할 수 없는 구문과 구조를 포함하지 않음을 검증함.	6.3.4.c
		코드 개발 중 소프트웨어 코드 표준 (예 : 복잡성 제한 및 코드 제약)을 준수하는지 검증함.	6.3.4.d
		상세 요구 사항이 소스 코드로 개발되었는지 검증함.	6.3.4.e
		소스 코드의 정확성 및 일관성을 결정하는 것을 검증함.	6.3.4.f
		소스코드 검증활동을 수행하지 않음.	

⑷ 통합

〈표〉 항공분야 안전개발 프로세스 조사 세부 항목 - 통합

설문 문항		선택항목	ref
22	개발된 소스코드를 통합하기 위해 수행하는 행위	오브젝트 코드와 실행 가능한 오브젝트 코드는 소스 코드와 데이터를 컴파일, 링크 및로드하여 생성함.	5.4.2.a
		모든 매개 변수 데이터 항목 파일을 생성함.	5.4.2.a
		소프트웨어 통합은 호스트 컴퓨터, 대상 컴퓨터 에뮬레이터 또는 대상 컴퓨터에서 수행함.	5.4.2.b
23	통합 프로세스 중 발견된 부적절한 입력에 대하여 유관 프로세스로 피드백하는 절차와 수행하는 방법	절차가 있으며, 절차에 따라 소프트웨어 요구 사항 프로세스, 소프트웨어 설계 프로세스, 소프트웨어 코딩 프로세스, 소프트웨어 계획 프로세스로 피드백 함.	5.2.2.g
		절차가 있으나 소프트웨어 요구 사항 프로세스, 소프트웨어 설계 프로세스, 소프트웨어 코딩 프로세스, 소프트웨어 계획 프로세스로 피드백하지 않음.	
		절차는 없으나, 소프트웨어 요구 사항 프로세스, 소프트웨어 설계 프로세스, 소프트웨어 코딩 프로세스, 소프트웨어 계획 프로세스로 피드백 함.	
		절차도 없고, 소프트웨어 요구 사항 프로세스, 소프트웨어 설계 프로세스, 소프트웨어 코딩 프로세스, 소프트웨어 계획 프로세스로 피드백하지 않음.	
24	통합 프로세스의 출력물을 검증하기 위해 수행하는 활동	통합 프로세스의 출력이 완전하고 올바른지 검증함. (데이터의컴파일,링크및로드및메모리맵에대한확인을 수행함을의미)	6.3.5a
		매개 변수 데이터 항목 파일은 상위 수준 요구 사항에 정의된대로 구조를 준수하는지검증함.	6.6.a
		매개 변수 데이터 항목 파일의 모든 요소(element)를	6.6.b

		검증함.	
		통합 프로세스에 대한 검증화동을 수행하지 않음.	
25	통합 프로세스 진행 중에 인증되어 사용되는 소프트웨어에 대한 패치가 필요한 경우 수행하는 행위	인증되어 제출 된 소프트웨어에는 패치를 사용하지 않음.	5.4.2.e
		알려진 컴파일러 문제와 같이 소프트웨어 개발 환경의 알려진 결함을 해결하기 위해 경우에 따라 제한적으로 사용함.	5.4.2.e
26	만약 패치를 사용할 경우 다음중에 어떤 활동	소프트웨어 구성 관리 프로세스가 패치를 효과적으로 추적 할 수 있는지 확인.	5.4.2.f
		패치 된 소프트웨어가 모든 적용 가능한 목표를 충족 시킨다는 증거를 제공하기위한 분석.	5.4.2.f
		패치 사용에 대한 소프트웨어 성취 요약의 올바른지 검토	5.4.2.f
		패치는 사용하나 위 활동은 수행하지 않음	

⑸ 테스트

〈표〉 항공분야 안전개발 프로세스 조사 세부 항목 - 테스트

설문 문항		선택항목	ref
27	테스트 설계시 수행하는 활동	특정 테스트 케이스는 정상 범위 테스트 케이스와 robustness (비정상 범위) 테스트 케이스를 포함하도록 개발되어야 함.	6.4.2.a
		특정 테스트 케이스는 소프트웨어 개발 프로세스에서 있는 소프트웨어 요구 사항과 고유한 오류 소스(error sources inherent) 로부터 개발되어야 함.	6.4.3.b
		테스트 절차는 테스트 케이스에서 생성됨	6.4.4.c
		테스트 설계 활동을 수행하지 않음.	
28	테스트 수행 환경	대상 컴퓨터에 로드 된 소프트웨어가 포함된 테스트 환경	6.4.1
		대상 컴퓨터 환경과 매우 유사한 환경	6.4.1
		테스트 환경은 고려하지 않음.	
29	테스트 케이스, 절차, 결과, 소프트웨어 요구사항과의 추적성을 확인하기 위해 수행하는 활동	소프트웨어 요구 사항과 테스트 케이스 간의 양방향 연관을 보여주는 추적 데이터가 개발되었습니다. 이 추적 데이터는 요구 사항 기반 테스트 범위 분석을 지원함.	6.5.a
		테스트 케이스와 테스트 절차 간의 양방향 연관을 보여주는 Trace Data 가 개발되었으며, 이 추적 데이터를 통해 테스트 케이스의 전체 세트가 테스트	6.5.b

		절차로 개발되었음을 확인할 수 있음.	
		테스트 절차와 테스트 결과 사이의 양방향 연관성을 보여주는 추적 데이터가 개발되었으며, 이 추적 데이터를 사용하면 전체 테스트 절차가 실행되었는지 확인할 수 있음.	6.5.c
30	요구 사항 기반 하드웨어 / 소프트웨어 통합 테스트를 위해 수행하는 활동	이 테스트 방법은 대상 컴퓨터 환경에서 작동하는 소프트웨어 및 상위 수준 기능에 관련된 오류 원본에 집중해야 함.	6.4.3.a
		요구 사항 기반 하드웨어 / 소프트웨어 통합 테스트를 통해 대상 컴퓨터의 소프트웨어가 상위 요구 사항을 충족 할 수 있음.	6.4.3.a
		요구 사항 기반 하드웨어 / 소프트웨어 통합 테스트를 수행하지 않음.	
31	요구 사항 기반 소프트웨어 통합 테스트 를 위해 수행하는 활동	이 테스트 방법은 소프트웨어 요구 사항 간의 상호 관계 및 소프트웨어 아키텍처에 의한 요구 사항 구현에 집중해야 함.	6.4.3.b
		요구사항 기반 소프트웨어 통합 테스트는 소프트웨어 구성 요소가 서로 올바르게 상호 작용하고 소프트웨어 요구 사항 및 소프트웨어 아키텍처를 충족시키는 지 확인함.	6.4.3.b
		요구 사항 기반 소프트웨어 통합 테스트 를 수행하지 않음.	
32	요구 사항 기반 low-level 테스트 를 위해 수행하는 활동	각 소프트웨어 구성 요소(컴포넌트)가 상세 요구 사항을 준수함을 입증하는 데 집중	6.4.3.c
		요구 사항 기반 low-level 테스트는 소프트웨어 구성 요소(component)가 상세 요구 사항을 충족시키는 지 확인	6.4.3.c
		요구 사항 기반 low-level 테스트 를 수행하지 않음.	
33	Normal test 기법 중에서 사용하는 기법	실수 및 정수 입력 변수는 valid equivalence classes 과 boundary values를 사용하여 실행함.	6.4.2.1.a
		시간 관련 함수(예:filters, integrators, and delays)의 경우 코드의 여러 반복을 수행하여 문맥에서 함수의 특성을 확인함.	6.4.2.1.b
		상태 전이(state transitions)의 경우 정상 작동 중에 가능한 전환을 실행하기 위한 테스트 케이스를 개발함.	6.4.2.1.c
		논리 방정식으로 표현 된 소프트웨어 요구 사항의 경우 일반 범위 테스트 케이스는 변수 사용 및 부울 연산자를 검증	6.4.2.1.d
34	Robustness test 기법 중에서 사용하는 기법	실수 및 정수 변수는 유효하지 않은 값의 등가 클래스(equivalence class selection of invalid values) 선택을 사용하여 실행하여,	6.4.2.2.a

		비정상적인 조건에서 시스템을 초기화를 확인	6.4.2.2.b
		incoming 데이터의 가능한 오류 모드(failure mode)를 결정해야 합니다. 특히 외부 시스템의 복잡한 디지털 데이터 열(complex, digital data strings)	6.4.2.2.c
		루프 카운트가 계산 된 값인 루프의 경우, 범위를 벗어난 루프 카운트 값을 계산하고 루프 관련 코드의 견고성을 입증하기 위해 테스트 케이스를 개발	6.4.2.2.d
		초과 된 프레임 시간에 대한 보호 메커니즘이 올바르게 응답하는지 확인하기 위한 점검이 이루어짐.	6.4.2.2.e
		필터, 적분기 및 지연과 같은 시간 관련 함수의 경우 연산 오버플로 방지 메커니즘을 위한 테스트 사례를 개발	6.4.2.2.f
		상태 전이의 경우, 소프트웨어 요구 사항에 의해 허용되지 않는 변환을 유발하기 위한 테스트 케이스가 개발	6.4.2.2.g
35	테스트 커버리지를 검증하기 위해 수행하는 활동	상위 수준 요구사항을 달성하는지 확인	6.4.4.a
		상세 요구사항을 달성하는지 확인	6.4.4.b
		적절한 커버리지 기준에 맞는 대한 소프트웨어 구조의 테스트 커버리지가 달성	6.4.4.c
		데이터 결합과 제어 결합을 포함한 소프트웨어 구조의 테스트 커버리지가 달성	6.4.4.d
		테스트 커버리지를 확인하는 활동을 하지 않음.	

2) 조사지에 들어갈 산출물 세부 내역

<표> 항공 분야 안전개발 프로세스에 정의된 산출물

No	DO-178C Outputs	산출물
1	Design Description	설계 설명
2	Executable Object Code	실행 가능한 객체 코드
3	Parameter Data Item File	매개 변수 데이터 항목 파일
4	PSAC(Plan for Software Aspects of Certification)	소프트웨어인증계획
5	Problem Reports	문제 보고

6	SCM Plan	소프트웨어형상관리 계획
7	SCM Records	소프트웨어형상관리 기록
8	SDP	소프트웨어개발계획
9	Software Accomplishment Summary	소프트웨어 성취도 요약
10	Software Configuration Index	소프트웨어 형상목록
11	Software Life Cycle Environment Configuration Index	소프트웨어 수명주기 환경 형상목록
12	Software Requirements Data	소프트웨어 요구사항 데이터
13	Software Verification Cases and Procedures	소프트웨어 검증 사례 및 절차
14	Software Verification Results	소프트웨어 검증 결과
15	Source Code	소스코드
16	SQA Plan	소프트웨어품질보증계획
17	SQA Records	소프트웨어품질보증기록
18	SVP	소프트웨어 검증계획
19	소프트웨어 Code Standards	소프트웨어 코드 표준
20	소프트웨어 Design Standards	소프트웨어 설계 표준
21	소프트웨어 Requirements Standards	소프트웨어 요구사항 표준
22	Trace Data	추적 데이터

5. 비표준 분야 세부 항목

1) 프로세스 조사 설계 세부 내용

〈표〉 비표준 분야 안전개발 프로세스 조사 세부 항목

구분	유형	실태 조사항목	선택 항목
소프트웨어 요구사항 명세	일반 활동	요구사항 명세를 위해 수행하는 활동	□시스템요구사항에서소프트웨어요구사항으로할당여부 □요구사항명세화및변경시문서반영여부 □요구사항명세책임자선임
	안전 활동	요구사항 명세 시 안전관련 수행 활동	□요구사항명세가SIL을만족할수있도록충분히상세한지 여부 □하드웨어와소프트웨어간의모든안전관련또는해당제약 사항을명세화 □안전요구사항명세는제품에필요한안전특성(안전기능/ 안전 무결성요구사항)을표현 □안전 무결성에대한평가를수행할수있도록,충분하게상세히작 성
		소프트웨어 안전 검증 계획 수립 관련 활동	□소프트웨어가안전요구사항을충족하는지입증하기위한 단계적계획수립 □계획수립에들어가는내용(시기/수행인력/환경/기법/합 격기준) □소프트웨어 안전확증을위한기술적전략(자동/수동,정적/동적,분석적/ 확률적) □확증계획에대한평가자의검토
	기법	안전성 관점에서 사용하는 요구분석 기법	□준공식적방법(Semi-formalmethods) □공식적인방법(Formalmethods) □시스템안전요구사항과소프트웨어 안전요구사항간의추적성 □안전요구사항관리소프트웨어사용
	산출물	안전 요구사항 명세 관련 작성하는 산출물 목록	□명세서(소프트웨어 안전요구사항) □계획서(소프트웨어기능안전검증) □소프트웨어 안전기록 □요구사항안전성분석보고서
소프트웨어 아키텍처	일반 활동	아키텍처 설계를 위해 수행하는 활동	□소프트웨어요구사항명세를기반으로소프트웨어아키텍 처명세를작성 □제안소프트웨어아키텍처를수립하고상세화 □모든하드웨어/소프트웨어상호작용을식별하고분석하 여상세하게작성

설계			□소프트웨어컴포넌트를식별 □소프트웨어와응용데이터/알고리즘간의상세한인터페이스를명시
	안전 활동	아키텍처 설계를 위해 수행하는 안전관련 활동	□안전기능설계/구현시하드웨어/시스템엔지니어협의 □시험가능성및변경용이성을고려한설계 □소프트웨어 안전/비안전기능간의독립성확보 □안전 무결성보증을위한제어흐름,데이터흐름모니터링가능여부
		안전 무결성을 유지할 명확한 아키텍처 표현법	보유 여부 - (표현법 선택지/주관식)
	기법	아키텍처 설계 시 사용하는 기법	□오류감지 □코드감지오류 □오류주장프로그래밍 □다양한모니터기술 □중복및가용성설계 □모듈방식 □소프트웨어 안전요구사항추적 □구조다이어그램 □컴퓨터지원사양및설계도구 □이벤트중심,최대응답시간보장 □정적리소스할당/동기화
	산출물	아키텍서 설계서에 기록되는 내용	□소프트웨어 안전요구사항명세를만족하는통합적기법과수단선택,그근거입증 □컴포넌트/서브시스템으로분할하는것에대한정보(신규여부,검증여부,안전관련성,안전 무결성) □소프트웨어/하드웨어상호작용결정,평가,기술 □명확한아키텍처표현법사용. □안전 무결성을유지할설계유형선택(설비입출력자료,통신자료,운영자인터페이스자료,유지보수자료,내부데이터베이스자료가포함) □안전요구사항명세를충족할수있도록적절한소프트웨어아키텍처통합시험명시
소프트웨어상세설계	일반 활동	소프트웨어 상세 설계를 위해 수행하는 활동	□소프트웨어 안전을위한요구사항명세,소프트웨어아키텍처설계,소프트웨어 안전을확증하기위한계획등의정보를상세설계시작전완료 □소프트웨어는모듈성,시험가능성,안전한변경을위한능력을갖추도록제작 □각소프트웨어모듈의설계와각소프트웨어모듈에적용되

			어야할시험방법을명시
	안전 활동	소프트웨어 안전기능을 확보를 위한 상세설계 요구사항 검토 항목	□시스템이안전한상태를달성유지할수있도록하는기능 □하드웨어에서결함을검출,알림,관리하는데관련된기능 □센서,액추에이터결함을검출,알림,관리하는데관련된기능 □소프트웨어스스로결함을검출,알림,관리하는데관련된기능(소프트웨어자가모니터링) □안전기능에대한온오프라인상의정기적시험에관련된기능 □비안전관련기능에대한인터페이스 □용량과응답시간성능 □소프트웨어와시스템간인터페이스
		안전 무결성 요구사항 충족을 보증할 단위/통합 시험 계획 수립	
		개발 사용 도구의 안전성 검증을 위해 검토하는 내용	□지원도구와프로그래밍언어 □사용하는프로그래밍언어가인증된제품인지 □사용하는프로그래밍언어가안전 무결성수준을만족하는지 □소스코드문서화정책,언어특성파악 □사용언어(프로그래밍)의확인된단점해소방안보유
	기법	안전 무결성 수준에 따른 설계를 위해 사용하는 기법	□구조화된메소드 □세미형식메소드 □정식디자인및개선방법 □컴퓨터지원설계도구 □방어프로그래밍 □모듈방식 □디자인및코딩표준 □구조화된프로그래밍 □소프트웨어 안전요구사항추적
		정형화된 모델링을 위해 사용하는 기법	□논리/기능블록다이어그램 □시퀀스다이어그램 □상태전이다이어그램 □데이터흐름도 □ERD □UML
	산출물	상세 설계 수행관련 산출물 보유 및 사용 산출물	□설명서(소프트웨어시스템설계) □명세서(소프트웨어모듈설계) □명세서(소프트웨어아키텍처통합시험,계획) □명세서(PE하드웨어와소프트웨어통합시험,계획) □명세서(소프트웨어시스템통합시험,계획) □명세서(소프트웨어모듈시험,계획)

			□지침서(개발도구와코딩매뉴얼) □소프트웨어설계안전성분석보고서
소프트웨어구현	일반활동	소프트웨어 구현를 위해 수행하는 활동	
	기법	각 소프트웨어 모듈의 소스코드에서 검토해야 할 것들	□오류가능성을줄이기위한코딩표준의사용 □동적객체/변수관련코딩표준 □제한된인터럽트사용관련코딩표준 □포인터의사용제한관련코딩표준 □제한된재귀사용관련코딩표준 □고수준언어의프로그램에서구조화되지않은제어흐름제거관련표준 □자동유형변환제거관련표준
소프트웨어테스트	일반활동	소프트웨어 시험를 위해 수행하는 활동	□각소프트웨어가소프트웨어설계단계에서명시된대로시험되고있음. □각소프트웨어가의도된기능이정상적으로작동되는지를 적절히시험함. □모든소프트웨어모듈/통합/시스템시험결과가기록됨 □시험후실패된내용에대한수정및검토활동이명시되어수행함
	안전활동	소프트웨어 안전기능을 확보를 위한 시험 단계 요구사항	단계별 기능안전 요건 시험 여부
	기법	모듈/통합 시험 기법	□확률론적테스트 □동적분석및테스트 □기능및블랙박스테스트 □성능테스트 □모델기반테스트 □인터페이스테스트 □테스트데이터기록및분석 □테스트관리및자동화도구사용 □소프트웨어 안전요구사항,시험계획간추적성 □정식검증
		하드웨어, 소프트웨어 통합 검토시 시 요구되는 기법	□기능및블랙박스테스트 □성능테스트 □하드웨어/소프트웨어통합테스트명세와소프트웨어요구사항간의추적
	산출물	통합 시험 명세에 포함되는 것	□시스템을통합단계에따라구분 □테스트케이스와시험데이터,시험의형태 □도구,지원소프트웨어,구성설명등의시험환경,완료판단기준
		시험 단계 수행관련 산출물 보유 및	□보고서(소프트웨어모듈시험) □보고서(소프트웨어모듈통합시험)

		사용하는 산출물	□보고서(소프트웨어시스템통합시험) □보고서(소프트웨어아키텍처통합시험) □보고서(PE하드웨어와소프트웨어통합시험) □보고서(소프트웨어기능안전검증)

2) 조사지에 들어갈 기법 및 산출물 세부 내역

<표> 비표준 분야 안전개발 프로세스에서 사용되는 기법

Category	Technique/Measure		SIL 1	SIL 2	SIL 3	SIL 4
Table A.1 - Software safety requirements specification	1a	준 공식적 방법	R	R	HR	HR
	1b	공식적인 방법	---	R	R	HR
	2	시스템 안전 요구 사항과 소프트웨어 안전 요구 사항 간의 전방 추적 성	R	R	HR	HR
	3	안전 요구 사항과 인식 된 안전 요구 사항 간의 역 추적 성	R	R	HR	HR
	4	위의 적절한 기술 / 조치를 지원하는 컴퓨터 지원 사양 도구	R	R	HR	HR
Table A.2 - Software design and development - software architecture design	1	오류 감지	---	R	HR	HR
	2	코드 감지 오류	R	R	R	HR
	3a	오류 주장 프로그래밍	R	R	R	HR
	3b	다양한 모니터 기술 (모니터와 모니터 기능이 동일한 컴퓨터에서 독립적 임)	---	R	R	
	3c	다양한 모니터 기술 (모니터 컴퓨터와 모니터링되는 컴퓨터가 분리되어 있음)	---	R	R	HR
	3d	다양한 중복성, 동일한 소프트웨어 안전 요구 사항 사양 구현	---	---	---	R
	3e	기능적으로 다양한 중복, 다른 소프트웨어 안전 요구 사항 사양 구현	---	---	R	HR
	3f	역방향 복구	R	R	---	NR
	3g	무국적 소프트웨어 설계 (또는 제한된 상태 설계)	---	---	R	HR
	4a	장애 복구 메커니즘 다시 시도	R	R	---	---
	4b	우아한 퇴화	R	R	HR	HR
	5	인공 지능 - 오류 수정	---	NR	NR	NR
	6	동적 재구성	---	NR	NR	NR

	7	모듈 방식	HR	HR	HR	HR
	8	신뢰할 수 있고 검증 된 소프트웨어 요소의 사용 (있는 경우)	R	HR	HR	HR
	9	소프트웨어 안전 요구 사항 사양과 소프트웨어 아키텍처 간의 포워드 추적 성	R	R	HR	HR
	10	소프트웨어 안전 요구 사항 사양과 소프트웨어 아키텍처 간의 역 추적 성	R	R	HR	HR
	11a	구조 다이어그램 **	HR	HR	HR	HR
	11b	세미 형식 메소드 **	R	R	HR	HR
	11c	정식 디자인 및 개선 방법 **	---	R	R	HR
	11d	자동 소프트웨어 생성	R	R	R	R
	12	컴퓨터 지원 사양 및 설계 도구	R	R	HR	HR
	13a	최대 사이클 시간을 보장하는 주기적 동작	R	HR	HR	HR
	13b	시간 트리거 아키텍처	R	HR	HR	HR
	13c	이벤트 중심, 최대 응답 시간 보장	R	HR	HR	-
	14	정적 리소스 할당	-	R	HR	HR
	15	공유 리소스에 대한 액세스의 정적 동기화	-	-	R	HR
Table A.3 - Software design and development - support tools and programming language	1	적절한 프로그래밍 언어	HR	HR	HR	HR
	2	강력한 형식의 프로그래밍 언어	HR	HR	HR	HR
	3	언어 하위 집합	---	---	HR	HR
	4a	공인 된 도구 및 공인 번역사	R	HR	HR	HR
	4b	도구 및 번역자 : 사용으로 인한 자신감 증가	HR	HR	HR	HR
Table A.4 - Software design and development - detailed design	1a	구조화 된 메소드 **	HR	HR	HR	HR
	1b	세미 형식 메소드 **	R	HR	HR	HR
	1c	정식 디자인 및 개선 방법 **	---	R	R	HR
	2	컴퓨터 지원 설계 도구	R	R	HR	HR
	3	방어 프로그래밍	---	R	HR	HR
	4	모듈 방식	HR	HR	HR	HR
	5	디자인 및 코딩 표준	R	HR	HR	HR
	6	구조화 된 프로그래밍	HR	HR	HR	HR

	7	신뢰할 수 있고 검증 된 소프트웨어 요소의 사용 (있는 경우)	R	HR	HR	HR
	8	소프트웨어 안전 요구 사항 사양과 소프트웨어 디자인 간의 포워드 추적 성	R	R	HR	HR
Table A.5 - Software design and development - software module testing and integration	1	확률 론적 테스트	---	R	R	R
	2	동적 분석 및 테스트	R	HR	HR	HR
	3	데이터 기록 및 분석	HR	HR	HR	HR
	4	기능 및 블랙 박스 테스트	HR	HR	HR	HR
	5	성능 테스트	R	R	HR	HR
	6	모델 기반 테스트	R	R	HR	HR
	7	인터페이스 테스트	R	R	HR	HR
	8	테스트 관리 및 자동화 도구	R	HR	HR	HR
	9	소프트웨어 설계 명세와 모듈 및 통합 테스트 명세 간의 전향 적 추적 성	R	R	HR	HR
	10	정식 검증	---	---	R	R
Table A.6 - Programmable electronics integration (hardware and software)	1	기능 및 블랙 박스 테스트	HR	HR	HR	HR
	2	성능 테스트	R	R	HR	HR
	3	하드웨어 / 소프트웨어 통합 및 하드웨어 / 소프트웨어 통합 테스트 사양에 대한 시스템과 소프트웨어 설계 요구 사항 간의 전향 적 추적 성	R	R	HR	HR
Table A.7 – Software aspects of system safety validation	1	확률 론적 테스트	---	R	R	HR
	2	공정 시뮬레이션	R	R	HR	HR
	3	모델링	R	R	HR	HR
	4	기능 및 블랙 박스 테스트	HR	HR	HR	HR
	5	소프트웨어 안전 요구 사항 사양과 소프트웨어 안전 유효성 검사 계획 간의 전향 추적 성	R	R	HR	HR
	6	소프트웨어 안전성 검증 계획과 소프트웨어 안전성 요구 사항 사양 간의 역 추적 성	R	R	HR	HR
Table A.8 - Modification	1	영향 분석	HR	HR	HR	HR
	2	변경된 소프트웨어 모듈 재확인	HR	HR	HR	HR
	3	영향을받는 소프트웨어 모듈 재확인	R	HR	HR	HR

	4a	전체 시스템 재 검증	---	R	HR	HR
	4b	회귀 검증	R	HR	HR	HR
	5	소프트웨어 구성 관리	HR	HR	HR	HR
	6	데이터 기록 및 분석	HR	HR	HR	HR
	7	소프트웨어 안전 요구 사항 사양과 소프트웨어 수정 계획 (재 검증 및 재 검증 포함) 간의 전향 적 추적 성	R	R	HR	HR
	8	소프트웨어 수정 계획 (재 검증 및 재 검증 포함)과 소프트웨어 안전 요구 사항 명세 사이의 역 추적 성	R	R	HR	HR
Table A.9 - Software verification	1	정식 증명	---	R	R	HR
	2	명세 및 디자인의 생기	R	R	R	R
	3	정적 분석	R	HR	HR	HR
	4	동적 분석 및 테스트	R	HR	HR	HR
	5	소프트웨어 설계 사양과 소프트웨어 검증 (데이터 검증 포함) 계획 간의 포워드 추적 성	R	R	HR	HR
	6	소프트웨어 검증 (데이터 검증 포함) 계획과 소프트웨어 설계 명세 간의 역 추적 성	R	R	HR	HR
	7	오프라인 수치 분석	R	R	HR	HR
		소프트웨어 모듈 테스트 및 통합	See Table A.5			
		프로그래밍 가능한 전자 통합 테스트	See Table A.6			
		소프트웨어 시스템 테스트 (검증)	See Table A.7			
Table A.10 - Functional safety assessment	1	점검표	R	R	R	R
	2	의사 결정 / 진리표	R	R	R	R
	3	고장 분석	R	R	HR	HR
	4	다양한 소프트웨어의 공통 원인 실패 분석 (다양한 소프트웨어가 실제로 사용되는 경우)	---	R	HR	HR
	5	신뢰성 블록 다이어그램	R	R	R	R
	6	8 절의 요구 사항과 소프트웨어 기능	R	R	HR	HR

		안전성 평가 계획 사이의 전방 추적 성				
Table B.1 - Design and coding standards	1	오류 가능성을 줄이기위한 코딩 표준의 사용	HR	HR	HR	HR
	2	동적 객체 없음	R	HR	HR	HR
	3a	동적 변수 없음	---	R	HR	HR
	3b	동적 변수 설치의 온라인 검사	---	R	HR	HR
	4	제한된 인터럽트 사용	R	R	HR	HR
	5	포인터의 사용 제한	---	R	HR	HR
	6	제한된 재귀 사용	---	R	HR	HR
	7	고수준 언어의 프로그램에서 구조화되지 않은 제어 흐름이 없음	R	HR	HR	HR
	8	자동 유형 변환 없음	R	HR	HR	HR
Table B.2 - Dynamic analysis and testing	1	경계 값 분석을 통한 테스트 케이스 실행	R	HR	HR	HR
	2	오류 추측에서 테스트 사례 실행	R	R	R	R
	3	오류 시드에서 테스트 사례 실행	---	R	R	R
	4	모델 기반 테스트 케이스 생성으로부터 테스트 케이스 실행	R	R	HR	HR
	5	성능 모델링	R	R	R	HR
	6	동등한 클래스와 입력 파티션 테스트	R	R	R	HR
	7a	구조 테스트 커버리지 (진입 점) 100 % **	HR	HR	HR	HR
	7b	구조 테스트 커버리지 (명세서) 100 % **	R	HR	HR	HR
	7c	구조 테스트 커버리지 (지점) 100 % **	R	R	HR	HR
	7d	구조 테스트 커버리지 (조건, MC / DC) 100 % **	R	R	R	HR
Table B.3 - Functional and black-box testing	1	원인 결과 다이어그램에서 테스트 사례 실행	---	---	R	R
	2	모델 기반 테스트 케이스 생성으로부터 테스트 케이스 실행	R	R	HR	HR
	3	프로토 타이핑 / 애니메이션	---	---	R	R
	4	경계 값 분석을 포함한 동등한 클래스와 입력 파티션 테스트	R	HR	HR	HR

	5	공정 시뮬레이션	R	R	R	R
Table B.4 – Failure analysis	1a	원인 다이어그램	R	R	R	R
	1b	이벤트 트리 분석	R	R	R	R
	2	결함 트리 분석	R	R	R	R
	3	소프트웨어 기능 장애 분석	R	R	R	R
Table B.5 - Modelling	1	데이터 흐름도	R	R	R	R
	2a	유한 상태 기계	---	R	HR	HR
	2b	공식적인 방법	---	R	R	HR
	2c	시간 페 트리 그물	---	R	HR	HR
	3	성능 모델링	R	HR	HR	HR
	4	프로토 타이핑 / 애니메이션	R	R	R	R
	5	구조 다이어그램	R	R	R	HR
Table B.6 - Performance testing	1	눈사태 / 스트레스 테스트	R	R	HR	HR
	2	응답 타이밍 및 메모리 제약	HR	HR	HR	HR
	3	성능 요구 사항	HR	HR	HR	HR
Table B.7 - Semi-formal methods	1	논리 / 기능 블록 다이어그램	R	R	HR	HR
	2	시퀀스 다이어그램	R	R	HR	HR
	3	데이터 흐름도	R	R	R	R
	4a	유한 상태 기계 / 상태 전이 다이어그램	R	R	HR	HR
	4b	시간 페 트리 그물	R	R	HR	HR
	5	엔터티 관련 특성 데이터 모델	R	R	R	R
	6	메시지 시퀀스 차트	R	R	R	R
	7	의사 결정 / 진리표	R	R	HR	HR
	8	UML	R	R	R	R
Table B.8 – Static analysis	1	경계 값 분석	R	R	HR	HR
	2	점검표	R	R	R	R
	3	제어 흐름 분석	R	HR	HR	HR
	4	데이터 흐름 분석	R	HR	HR	HR
	5	오류 추측	R	R	R	R
	6a	특정 기준을 포함한 공식 검사	R	R	HR	HR
	6b	워크 쓰루 (소프트웨어)	R	R	R	R
	7	상징적 실행	---	---	R	R
	8	디자인 검토	HR	HR	HR	HR
	9	런타임 오류 동작의 정적 분석	R	R	R	HR
	10	최악의 실행 시간 분석	R	R	R	R
Table B.9 - Modular approach	1	소프트웨어 모듈 크기 제한	HR	HR	HR	HR
	2	소프트웨어 복잡성 제어	R	R	HR	HR
	3	정보 숨기기 / 캡슐화	R	HR	HR	HR
	4	매개 변수 개수 제한 / 고정 된 서브	R	R	R	R

		프로그램 매개 변수 수				
	5	서브 루틴과 함수에서 한 개의 엔트리 포인트 / 한 개의 출구 포인트	HR	HR	HR	HR
	6	완전히 정의 된 인터페이스	HR	HR	HR	HR

〈표〉 비표준 분야 안전개발 프로세스에서 산출물

단계명	IEC 61508 Part1 분류
소프트웨어개발계획	설명서(전체적인개념) 설명서(전체적인범위정의) 설명서(위험원및리스크분석)
	설명서(전체적인안전요구사항할당) 명세서(전체적인안전요구사항)
	계획서(소프트웨어 기능안전)
소프트웨어요구분석	명세서(소프트웨어 안전 요구사항)
	계획서(소프트웨어 기능안전 검증)
소프트웨어설계	설명서(소프트웨어 아키텍처 설계)
	설명서(소프트웨어 시스템 설계)
	명세서(소프트웨어 모듈 설계)
	명세서(소프트웨어 아키텍처 통합 시험, 계획)
	명세서(PE 하드웨어와 소프트웨어 통합 시험, 계획)
	명세서(소프트웨어 시스템 통합 시험, 계획)
	명세서(소프트웨어 모듈 시험, 계획)
	지침서(개발도구와코딩매뉴얼)
소프트웨어구현	보고서(코드 검토)
소프트웨어통합	보고서(소프트웨어 모듈 시험)
	보고서(소프트웨어모듈통합시험) 보고서(소프트웨어시스템통합시험) 보고서(소프트웨어아키텍처통합시험) 보고서(PE하드웨어와소프트웨어통합시험)
	보고서(소프트웨어 기능안전 검증)
소프트웨어운영	요청서(소프트웨어 변경)

	요청서(소프트웨어 폐기)
	지침서(사용자)
	지침서(운영과 유지보수)
	지침서(소프트웨어 변경 절차)
	보고서(소프트웨어 변경 영향 분석)
	일지(소프트웨어 변경)
공통	계획서(소프트웨어 확인)
	보고서(소프트웨어 확인)
	계획서(소프트웨어 기능안전성 평가)
	보고서(소프트웨어 기능안전성 평가)

Ⅱ. 용어 정리

- ISO26262 : 자동차 기능 안전 표준 (Automotive Functional Safety Standard)
- FSM : Finite-state machine
- IATF16949 : International Automotive Task Force (ISO 9001 규격을 기본으로 하여 자동차 업계의 추가적인 시스템 요구사항 포함)
- A-SPICE : Automotive Software Process Improvement & dEtermination
- CMMi : Capability Maturity Model Integration

- IEC 61508 : Functional Safety of Electrical/Electronic/Programmable Electronic Safety-related Systems
- EN50126 : Railway RAMS(Reliablity, Availablity, Maintainability and. Safety) standard
- EN50128 : Railway control and protection systems
- EN50129 : Railway Communication, signalling and processing systems
- IEC62279 : EN50128
- IEC62425 : EN50129
- IEC62278 : EN50126
- IEC61511 : Functional safety - Safety instrumented systems for the process industry sector
- IEC/EN 62061 : Functional safety of electrical, electronic and programmable electronic control systems

- RTCA DO-173 : Minimum Operational Performance Standards for Airborne Weather and Ground Mapping Pulsed Radars
- RTCA DO-178C : Software Considerations in Airborne Systems and Equipment Certification
- DO254(RTCA/DO-254) : Design Assurance Guidance for Airborne Electronic Hardware
- MIL-HDBK-516C : AIRWORTHINESS CERTIFICATION CRITERIA
- AS9100D : Quality Management Systems

- ISO 9100 : standard for types of vacuum lug finishes for glass containers
- MIL-STO-882E : DEPARTMENT OF DEFENSE STANDARD PRACTICE

- IEC와 항행통신장비표준
- ISO 9001 : Quality management
- ISO 140001 : Standards related to environmental management
- ASME : American Society of Mechanical Engineers
- IMO SQA : International Maritime Organization - Software Quality Assurance guidelines for e-navigation systems

참 고 문 헌

[국내 문헌]

강해달 (2012). 『소프트웨어 테스팅 관점에서 본 소프트웨어 기능안전성 표준 비교』. 정보과학회

권기현 (2012), 『고 신뢰 융합소프트웨어를 위한 필수 소프트웨어공학기술 조사 용역』,정보통신산업진흥원

권기춘 외(2017), 『원전 안전 소프트웨어 위해도 및 안전보증 평가기술 개발』, 한국원자력연구원

김건명 (2016). 『IEC 61508에 기반한 원자력 발전소용 안전 등급 제어기의 SIL 분석에 대한 사례 연구』. 신뢰성응용연구

김기영, 고병각, 장중순, 천성일 (2010). 『플로우차트 기반 안전 무결성수준 평가 절차』. 신뢰성응용연구

김성규, 김용수 (2014). 『기능안전을 위한 IEC 61508의 안전수명주기에 관한 연구』. 신뢰성응용연구

김우식외6 (2001), 『소프트웨어_품질_인증을_위한_참조_모델』, 정보과학회

노경현, 이민재, 이금석 (2013). 『ISO 26262와 CMMI 통합 프레임웍을 활용한 효율적인 차량 기능안전성 평가』. 한국정보과학회

송재효, 박상구, 김영상 (2016). 『PSD의 안전성 및 신뢰성 확보를 위한 요구사항 및 안전관리 프로세스의 적용 방안』. 한국철도학회

심규돈, 이종우, 박채영, 김재남, 서영준 (2010). 『IEC 62279 요구사항 충족을 위한 열차제어시스템의 소프트웨어 품질보증 활동에 관한 연구』. 한국철도학회

옥승민 (2018). 『산업용 소프트웨어 안전성 확보의 중요성과 관련 표준 동향』. 전력전자학회

우경일, 석민진 (2013). 『ISO 26262 에서 요구하는 안전 활동 관리 (Safety Activity Management) 방안 연구』. 전자공학회

윤성현, 김연준, 최윤자, 김진삼, 안성호 (2009). 『차량용 안전관련 소프트웨어 개발을 위한 국제표준과 안전성 요구사항에 대한 연구』. 한국자동차공학회

윤원근, 이백준, 진영권 (2012). 『항공안전을 위한 소프트웨어 및 하드웨어 품질 인증 기술 비교분석 연구』. 한국항공우주학회

윤원식 (2013). 『DO-178B와 CMMISPICE기반의 소프트웨어 수명주기 프로세스 비교』. 한국항공우주학회
윤학선, 이기서, 이종우, 박재영 (2009). 『도시형자기부상열차 열차제어시스템 안전성 평가 인증 체계』. 한국철도학회
이승곤, 한무희, 변태선 (2014). 『ISO26262 기능안전 적용을 위한 소프트웨어 V&V 이행 전략 연구』. 한국자동차공학회
이영준, 김장열, 차경호, 천세우, 이장수, 권기춘 (2007). 『철도 안전 소프트웨어를 위한 개발 기준 연구』. 한국철도학회
임상우, 이서정, 양회석 (2017). 『항해장비 소프트웨어 기능안전성 확보를 위한 위험분석 단계 연구』. 한국디지털콘텐츠
조진희 (2014). 『ISO 26262의 소프트웨어레벨 안전성 분석 달성 방안』. 오토저널
조진희, 박경민, 한태만, 정양재, 전서현, 김현수 (2009). 『자동차 기능 안전성 표준 ISO 26262 분석과 적용방안 연구』. 한국자동차공학회
조진희, 성기순, 한태만, 김현수 (2013). 『자동차 기능 안전성 표준 ISO 26262의 소프트웨어레벨 개발 적합성 달성 방안』. 전자공학회
조현명 (2016). 『DO-178C에 따른 항공용 소프트웨어의 적합성 입증 방법에 관한 연구』. 항공진흥학회
진회승(2017), 『자동차 산업의 SW안전 이슈와 해결과제』. SPRi
진회승(2017). 『CPS의 기능, 비기능적 SW요구사항 분석』. SPRi
최성호 (2012). 『ISO 26262 대응 Process 구축 문제점 및 현실적 대응 방안』. 한국자동차공학회
황종규(2017) , 『철도분야에서의 SW안전』, 2017 SW안전 국제컨퍼런스
황종규, 조현정, 신승권, 정락교 (2009). 『열차제어시스템 안전성 활동 기술체계의 국내 적용방안』. 대한전기학회
정보통신산업진흥원 (2017), 『공통 분야(IEC 61508) 소프트웨어신뢰·안전성 확보를 위한 가이드』
정보통신산업진흥원 (2017), 『자동차 분야(ISO26262) 소프트웨어신뢰·안전성 확보를 위한 가이드』
정보통신산업진흥원 (2017), 『철도 분야(IEC62279) 소프트웨어신뢰·안전성 확보를 위한 가이드』
한국소프트웨어진흥원 (2005), 『시스템 수명주기 프로세스 프레임워크 보고서』
소프트웨어정책연구소(2015,2017,2018), 소프트웨어안전 산업동향 조사
Byung Chul Kim, Young Jin Kim (2012). 『Case Study on the Assessment of SIL Using FMEDA』.

산업공학

KS R ISO 26262-6 (2015), Road vehicles - Functional safety - Part 6: Product development at the software level

[국내 사이트]

산업안전보건공단, http://www.kosha.or.kr/

[해외 문헌]

Barbara J. Czerny et al.(2003), "A Software Safety Process for Safety-Critical Advanced Automotive Systems", PROCEEDINGS of the 21st INTERNATIONAL SYSTEM SAFETY CONFERENCE

DOE_Guide_414_1_4 (2005),"SAFETY SOFTWARE GUIDE for USE with 10 CFR 830 Subpart A, Quality Assurance Requirements, and DOE O 414.1C, Quality Assurance "

ITER (2013), "SEQA-45 - Software Engineering and Quality Assurance"

Kesseler, Ernst (2008), "Safe software certification, Safety is no accident DO-178B", University of Groningen

Myron Hecht(2009), Aerospace Corp., member of AADL & DO-178C committee, "Automated safety analysis of several satellite systems for JPL"

Nancy.G. Leveson, C.S. Turner(1993), "Medical devices The Therac25", IEEE

Nancy G. Leveson et al.(2018) , STPA Handbook

NHTSA(National Highway Traffic Safety Administration, 2015), Report to Congress: "Electronic Systems Performance in Passenger Motor Vehicles"

NTSB(2018), "Preliminary Report Highway: HWY18MH010"

Peter Feiler, "Challenges in Existing System Safety Practices", CMU, Software Solutions Symposium 2017

Peter Feiler, "Virtual System Integration and Verification", CMU, Software Solutions Symposium 2017

Raghu Singh, "INTERNATIONAL STANDARD ISO/IEC 12207 SOFTWARE LIFE CYCLE

PROCESSES" , Federal Aviation Administration

RTCA DO-178C (2011), "Software Considerations in Airborne Systems and Equipment Certification"

Smith, David J. and Simpson, Kenneth G. L. (2004), "Functional Safety(A Straightforward Guide to Applying IEC 61508 and Related Standards)", Hutterwirth-Heinemann

[해외 사이트]

61508 협회, http://www.61508.org

주 의

1. 이 보고서는 소프트웨어정책연구소에서 수행한 연구보고서입니다.
2. 이 보고서의 내용을 발표할 때에는 반드시 소프트웨어정책연구소에서 수행한 연구결과임을 밝혀야 합니다.

95000
9 788961 084352
ISBN 978-89-6108-435-2 (PDF)