


1단계(3차년도) 주요 결과물

(과제명) 대규모 분산 에너지 저장장치 인프라의 안전한 자율운영
및 성능 평가를 위한 지능형 SW 프레임워크 개발

(과제번호) 2021-0-00077

- 결과물명 : 안전 SW 검증 지침서 v1.0
- 작성일자 : 2023년 11월 17일

과학기술정보통신부 SW컴퓨팅산업원천기술개발사업
“1단계(3차년도) 주요 결과물” 로 제출합니다.

수행기관	성명/직위	확인
슈어소프트테크(주)	심정민/상무이사	

정보통신기획평가원장 귀하

안전 SW 검증 지침서 v1.0

SURESOFT

2023년 09월 20일

슈어소프트테크(주)

<제목 차례>

1. SW 생명 주기 별 검증 활동	5
1.1. SW 검증 계획 작성	7
1.2. SW 설계 검증	8
1.2.1. SW 요구사항 검증	9
1.2.2. SW 상세 설계 검증	10
1.3. SW 구현 검증	12
1.3.1. 소스코드 검증	13
1.3.2. 추적성 검증	14
1.4. SW 모듈 시험	14
1.4.1. 모듈 단위 시험	14
1.5. SW 통합 시험	17
1.5.1. 모듈 통합 시험	17
1.6. 시스템 통합 시험	18
1.6.1. SW 시스템 통합 시험	18
2. SW V&V 보고	21
2.1. V&V 보고요건	21
3. V&V 관리 절차	23
3.1. 관리공정	23
3.1.1. V&V 관리 활동	23
4. 결론	25

<표 차례>

표 1. 단계별 검증 항목 기반 KC 62619 수행 여부	6
표 2. SW 설계 검증 - 검증 지표	8
표 3. 추적성 분석 매트릭스	10
표 4. SW 구현 검증 - 검증 지표	12
표 5. 코딩 규정	13
표 6. 소스코드 메트릭(Metrics) 종류	14
표 7. SW 모듈 시험 지표	15
표 8. SW 통합 시험 지표	17
표 9. 시스템 통합 시험 검증 지표	19
표 10. 소프트웨어 V&V 체크리스트 - IEEE 1471	26
표 11. 소프트웨어 V&V 체크리스트 - IEEE 830	34
표 12. 소프트웨어 V&V 체크리스트 - IEEE 1016	42
표 13. 코딩 규칙 - MISRA-C:2012	51
표 14. 코딩 규칙 - ISO 26262	58
표 15. 코딩 규칙 - NUREG/CR-6463	59
표 16. 코딩 규칙 - CWE_C	65
표 17. 코딩 규칙 - IEC 62279	67

<그림 차례>

그림 1. ESS SW 개발 및 검증 프로세스	5
그림 2. ESS SW 개발 및 검증 프로세스 단계별 검증 대상	5
그림 3. SW 검증 주요 공정	8
그림 4. SW 요구사항 검증 수행 절차	9
그림 5. SW 상세 설계 검증 수행 절차	11

<부록 차례>

부록 A. 체크리스트	26
부록 B. 참조 문서	50
부록 C. 코딩 규칙	51

1. SW 생명 주기 별 검증 활동

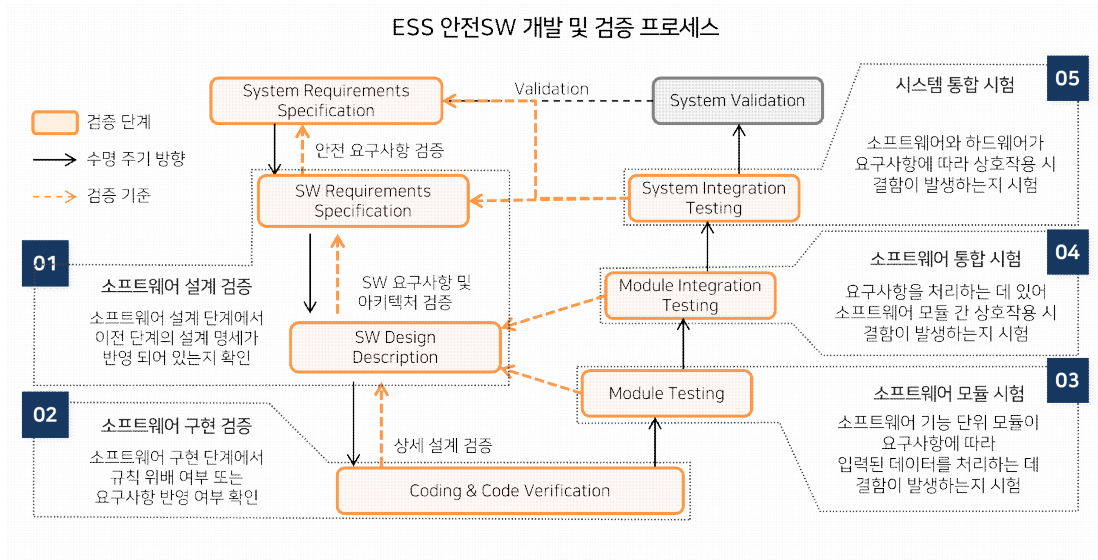


그림 1. ESS SW 개발 및 검증 프로세스

그림 1의 ESS SW 개발 및 검증 프로세스는 IEC 61508 기반으로 만들어진 것으로, SW Requirements Specification은 IEC 61508에서 정의하는 Software Architecture에 관한 내용을 일부 포함하며, SW Design Description은 IEC 61508에서 정의하는 Software Architecture/System/Module Design을 포함한다.

아래 그림 2는 ESS SW 개발 및 검증 프로세스의 단계별 검증 대상을 보여준다.

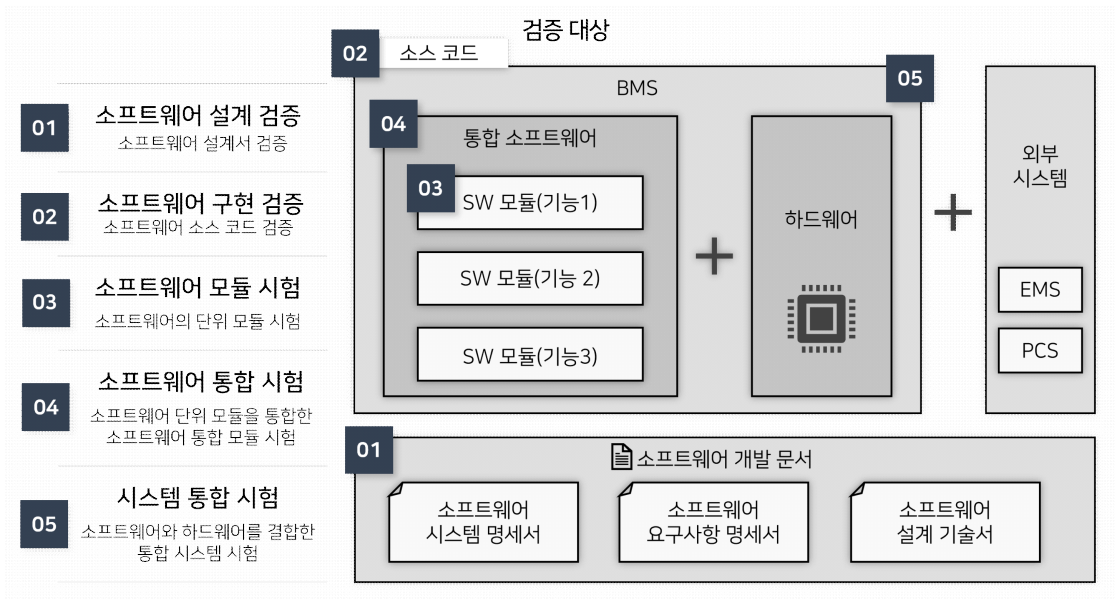


그림 2. ESS SW 개발 및 검증 프로세스 단계별 검증 대상

ESS SW 개발 및 검증 프로세스 단계별 검증 관련 설명은 다음과 같다.

- ① SW 설계 검증 : 상위 시스템 요구사항(상위 기술 사양서, 사용자 매뉴얼 등)으로부터 SW 요구사항이 올바르게 도출되었는지를 확인하는 과정을 시작으로 설계 검증을 수행하며 해당 단계에서는 SW 요구사항 명세서와 상세 설계 기술서에 대한 문서 구성 및 추적성 검증을 수행한다.
- ② SW 구현 검증 : 설계 검증 단계에서 산출된 설계 문서를 기반으로 SW 구현 단계에서 산출된 소스 코드에 대해 코딩 규칙 및 요구사항 기반 구현 검증을 수행한다.
- ③ SW 모듈 시험 : SW 설계 기술서에 기술된 각 모듈들을 대상으로 SW 모듈 시험을 수행한다. 분기가 존재하는 코드에 대한 모듈 검증 시에는 커버리지 요구사항이 추가되며 커버리지 요구사항은 분기 커버리지 100%를 기준으로 하되 경우에 따라 협의 하에 조정될 수 있다.
- ④ SW 통합 시험 : SW 단위 모듈을 기능 단위 또는 모듈 간 상호 연계 확인을 위한 적절한 단위로 통합하여 SW 통합 모듈 시험을 수행한다.
- ⑤ 시스템 통합 시험 : 전체 SW와 HW를 결합한 통합 환경에서 시스템 단위의 기능 시험을 수행한다.

SW 개발 및 검증 프로세스의 검증 단계별 검증 항목과 각 항목에 맞게 검증 지표를 도출하였다. 또한, 추가로 도출된 검증 항목이 KC 62619에서 수행되고 있는지 비교하였다.

단계 별 검증 항목				
검증 단계	검증 항목	설명	검증 지표	KC 62619
SW 설계 검증	SW 요구사항 검증	<ul style="list-style-type: none"> SW 요구사항 명세서 구성 적합성 검증 SW 요구사항과 시스템 요구사항간 추적성 검증 	적합성 검증 체크리스트 및 추적성 커버리지	X
	SW 상세 설계 검증	<ul style="list-style-type: none"> SW 설계 기술서 구성 적합성 검증 SW 설계 기술서 아키텍처 설계 적합성 검증 SW 설계 요소와 SW 요구사항 추적성 검증 	적합성 검증 체크리스트 및 추적성 커버리지	X
SW 구현 검증	소스코드 검증	<ul style="list-style-type: none"> 소스코드가 코딩 표준을 준수하는지 검증 소스코드와 SW 설계 요구사항 추적성 검증 	규칙 위반 개수 추적성 커버리지	X
SW 모듈 시험	모듈 단위 시험	SW 설계 기술서의 기능/비기능 요구사항 충족여부 확인 시험	요구사항 커버리지	X
		기능/비기능 시험을 통해 소스 코드 상에서 시험 수행된 코드의 커버리지를 검증	코드 커버리지	
SW 통합 시험	모듈 통합 시험	SW 요구사항 명세서의 기능/비기능 요구사항을 충족하는지 시험	요구사항 커버리지	X
		모듈 간 인터페이스에 해당하는 함수 간 호출의 코드 커버리지를 검증	함수 호출 커버리지	
시스템 통합 시험	시스템 기능 시험	시스템의 안전기능 요구사항을 충족하는지 시험	기능 요구사항 커버리지	△ KC 62619 8.2 안전기능시험에 일부 포함
	시스템 비기능 시험	시스템(SW)의 비기능 요구사항을 충족하는지 시험	비기능 요구사항 커버리지	X
	시스템 강건성 시험	악의 조건에서 시스템 무정지 강건성을 확인하는 시험	강건성 요구사항 커버리지	△ 부속서 D.5에서 안전기능 기능시험 수행

표 1. 단계별 검증 항목 기반 KC 62619 수행 여부

표 1에 기술된 각 검증 항목별 상세 설명과 검증 지표에 대한 설명은 각 검증 단계별 활동을 설명하는 4.2절 ~ 4.6절에서 확인할 수 있다.

1.1. SW 검증 계획 작성

SW에 대한 검증계획서 작성은 SW 검증계획서에 포함되어야 할 내용들에 대한 요구사항들을 규정하고 있는 IEEE Std. 1012-2004를 참고하여 ESS SW 검증에 적합하도록 구성하였다. SW 검증계획서의 내용은 아래와 같이 작성할 수 있으며, 예시의 목차는 일반적인 사항들로 구성되어 있으며 필요한 경우 SW 개발 및 검증 프로세스에 맞게 목차를 조정할 수 있다.

SW 검증계획서 목차
1. 목적
2. 참고자료
3. 용어 정의
4. V&V 개요
4.1 조직
4.2 종합 일정
4.3 SW 무결성 등급 체계
4.4 자원 요약
4.5 책임 사항
4.6 도구, 기법 및 방법
5. V&V 공정
5.1 검증 계획
5.2 설계 검증
5.2.1 요구사항 검증
5.2.2 상세 설계 검증
5.3 구현 검증
5.3.1 소스코드 검증
5.3.1.1 기능 안전 검증
5.3.1.2 보안 취약점 점검
5.3.1.3 소스코드 메트릭(Metric)
5.3.2 추적성 검증
5.4 SW 모듈 시험
5.4.1 모듈 단위 시험
5.5 SW 통합 시험
5.5.1 모듈 통합 시험
5.6 시스템 통합 시험
5.6.1 SW 시스템 통합 시험
6. V&V 보고요건
6.1 업무보고서
6.2 활동 요약보고서
6.3 이상보고서
6.4 V&V 최종보고서
6.5 선택적 V&V 보고서
7. V&V 관리 절차

각 공정을 단계별로 나타내면 아래 그림과 같다.

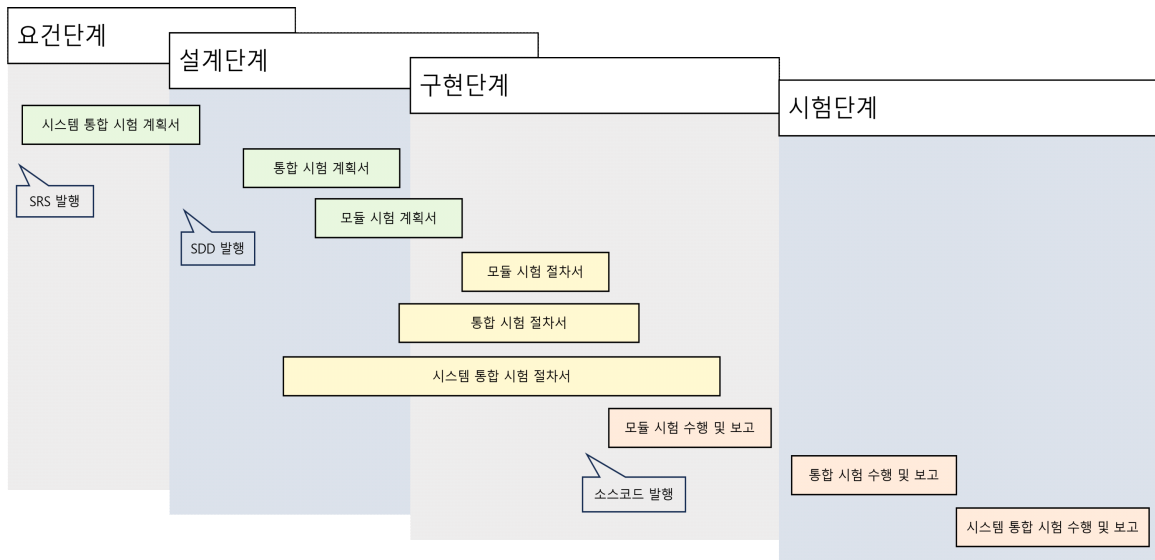


그림 3. SW 검증 주요 공정

1.2. SW 설계 검증

SW 설계 검증은 기본적으로 SW 생명 주기 중 발생하는 설계문서들에 대한 품질 향상이 목적이며 이를 위해 각 설계 문서에 대한 표준 기반의 객관적인 평가와 상위 문서와의 추적성 검증이 수행되어야 한다.

각 수준별 추적성 검증 수행 및 시험 문서 작성에는 SW 확인 및 검증의 대표적인 국제 표준인 IEEE Std. 1012-2004에서 기술하고 있는 방법을 적용하며 각 문서별 구성과 내용의 충분성은 IEEE Std. 830, 1016을 기반으로 확인된다.

SW 설계 검증 단계에서 수행되는 검증 항목들에 대한 검증 지표는 아래와 같다.

SW 설계 검증 : 해당 설계 문서가 상위 설계 문서에 기술된 요구사항을 충족하는지 확인				KC 62619 수행 여부
검증 항목	설명	검증 지표	지표 산정 방법	
SW 요구사항 검증	SW 요구사항 명세서가 표준에서 요구하는 항목들을 갖추고 있는지 확인	추적성 커버리지	시스템 요구사항과 추적된 SW 요구사항 개수 SW 요구사항 개수	X
	SW 요구사항이 시스템 요구사항의 내용을 기반으로 작성되었는지 확인	적합성 검증 체크리스트	적합성 검증 체크리스트 만족/불만족 항목 개수 전체 체크리스트 항목 개수	X
SW 상세 설계 검증	SW 설계 기술서가 표준에서 요구하는 항목들을 갖추고 있는지 확인	적합성 검증 체크리스트	적합성 검증 체크리스트 만족/불만족 항목 개수 전체 체크리스트 항목 개수	X
	SW 설계 기술서가 표준에서 요구하는 아키텍처 설계 항목들을 갖추고 있는지 확인	적합성 검증 체크리스트	적합성 검증 체크리스트 만족/불만족 항목 개수 전체 체크리스트 항목 개수	X
	SW 설계 기술서가 SW 요구사항 명세서의 내용을 기반으로 작성되었는지 확인	추적성 커버리지	SW 요구사항과 추적된 SW 설계 요소 개수 SW 설계 요소 개수	X

표 2. SW 설계 검증 - 검증 지표

세부 검증 방법 및 절차는 각 항목별 절 내부에 기술되어 있다.

1.2.1. SW 요구사항 검증

SW 요구사항 검증은 다음 두 가지 방식으로 진행된다.

- 1) 표준 기반 문서 적합성 검증
- 2) 추적성 검증

SW 요구사항 검증 수행을 위한 절차는 아래와 같다.



그림 4. SW 요구사항 검증 수행 절차

1.2.1.1. 표준 기반 문서 적합성 검증

표준 기반 문서 적합성 검증은 SW 요구사항 명세서 작성을 위한 국제 표준인 IEEE Std. 830을 기반으로 문서의 구성과 내용이 충분한지 체크리스트 기반 검증을 수행하는 방식이다. IEEE Std. 830은 SW 요구사항을 문서화하기 위한 내용을 기술한 표준으로 SW 요구사항 명세서를 명확하고 체계적으로 기술하기 위한 가이드라인을 제공한다.

검증 담당자는 부록 A의 IEEE Std. 830 검증 체크리스트를 활용하여 요구사항 명세서가 다음 항목들을 만족하는지 확인하게 된다.

- 정당성 (Correctness): 각각의 SW 요구사항들과 연관된 시스템 요구사항과의 관계가 올바른지 검증
- 일관성 (Consistency): SW와 시스템 요구사항의 상세한 정도가 일관성 있게 명시되어 있는지 확인
- 완결성 (Completeness): 모든 SW 요구사항이 시스템 요구사항과 일치함을 입증할 수 있을 만큼 상세한 수준으로 추적할 수 있는 확인

- 정확성 (Accuracy): 시스템 성능 및 운영 특성이 SW 요구사항에 정밀하게 명시되어 있는지 확인
- 가독성(Readability): 문서가 읽기 쉽고, 이해 가능하며, 의도한 대상자에게 모호하지 않은지 확인
- 시험가능성(Testability): 각 SW 설계 명세 및 시스템 설계를 검증하기 위한 객관적인 허용 기준이 있는지 확인

이 과정에서 발견된 이상(anomaly)은 이상보고서를 통해 보고한다.

1.2.1.2. 추적성 검증

추적성 검증은 상위 시스템 요구사항으로부터 SW 요구사항이 올바르게 도출되었는지 정확성 일관성 관점에서 검증하고 추가된 SW 요구사항들이 시스템 요구사항에 반하지 않는지, 정밀하게 명시되어 있는지 검증하는 방식이다.

추적성 검증은 아래 4가지 특성들을 기준으로 수행한다.

- 정당성 (Correctness): 각각의 SW 요구사항들과 연관된 시스템 요구사항과의 관계가 올바른지 검증
- 일관성 (Consistency): SW와 시스템 요구사항의 상세한 정도가 일관성 있게 명시되어 있는지 확인
- 완결성 (Completeness): 모든 SW 요구사항이 시스템 요구사항과 일치함을 입증할 수 있을 만큼 상세한 수준으로 추적할 수 있는 확인
- 정확성 (Accuracy): 시스템 성능 및 운영 특성이 SW 요구사항에 정밀하게 명시되어 있는지 확인

검증 담당자는 시스템 요구사항과 SW 요구사항의 관계를 위 특성들의 관점을 고려하여 분석한 후 아래와 같은 형태의 추적성 분석 매트릭스를 활용하여 추적성 분석 결과를 정리한다. 이 과정에서 발견된 이상(anomaly)은 이상보고서를 통해 보고한다.

SW 설계 검증: SW 요구사항 추적성 검증						
시스템 요구사항 문서			SW 요구사항 문서			세부 검증 결과
절번호	ID	내용	절번호	ID	내용	

표 3. 추적성 분석 매트릭스

1.2.2. SW 상세 설계 검증

SW 상세 설계 검증은 다음 두 가지 방식으로 진행된다.

- 1) 표준 기반 문서 적합성 검증
- 2) 추적성 검증

SW 상세 설계 검증 수행을 위한 절차는 아래와 같다.



그림 5. SW 상세 설계 검증 수행 절차

1.2.2.1. 표준 기반 문서 적합성 검증

표준 기반 문서 적합성 검증은 SW 상세기술서가 IEEE Std. 1016과 IEEE Std. 1471를 기반으로 문서의 구성과 내용이 기능 및 비기능 요구사항을 구현하는데 충분한지 체크리스트 기반 검증을 수행하는 방식이다. IEEE Std. 1016과 IEEE Std. 1471는 SW 상세기술서를 명확하고 체계적으로 기술하기 위한 가이드라인을 제공한다.

검증 담당자는 부록 A의 IEEE Std. 1016과 IEEE Std. 1471 검증 체크리스트를 활용하여 상세기술서가 다음 항목들을 만족하는지 확인하게 된다.

- **정당성 (Correctness):** 각각의 SW 요구사항들과 연관된 시스템 요구사항과의 관계가 올바른지 검증
- **일관성 (Consistency):** SW와 시스템 요구사항의 상세한 정도가 일관성 있게 명시되어 있는지 확인
- **완결성 (Completeness):** 모든 SW 요구사항이 시스템 요구사항과 일치함을 입증할 수 있을 만큼 상세한 수준으로 추적할 수 있는 확인
- **정확성 (Accuracy):** 시스템 성능 및 운영 특성이 SW 요구사항에 정밀하게 명시되어 있는지 확인
- **가독성(Readability):** 문서가 읽기 쉽고, 이해 가능하며, 의도한 대상자에게 모호하지 않은지 확인
- **시험가능성(Testability):** 각 SW 설계 명세 및 시스템 설계를 검증하기 위한 객관적인

허용 기준이 있는지 확인

이 과정에서 발견된 이상(anomaly)은 이상보고서를 통해 보고한다.

1.2.2.2. 추적성 검증

추적성 검증은 SW 요구사항으로부터 SW 설계 요구사항이 올바르게 도출되었는지 정당성, 일관성, 완결성, 정확성 관점에서 검증하고 추가된 SW 설계 요구사항들이 SW 요구사항에 반하지 않는지 검증한다.

추적성 검증은 아래 4가지 특성들을 기준으로 수행한다.

- 정당성 (Correctness): 각 설계 요구사항과 SW 요구사항 간의 관계성을 검증
- 일관성 (Consistency): 설계 요구사항과 SW 요구사항 간의 관계성이 일관성 있는 수준으로 명시되어 있는지 확인
- 완결성 (Completeness): 모든 설계 요소가 SW 요구사항으로부터 추적 가능한지 그리고 모든 SW 요구사항이 설계 요소에 대해 추적 가능한지 확인
- 정확성 (Accuracy): 시스템 성능 및 운영 특성이 SW 요구사항에 정밀하게 명시되어 있는지 확인

SW 요구사항과 SW 설계 요소의 관계를 위 특성들의 관점을 고려하여 분석한 후 표 3와 같은 형태의 추적성 분석 매트릭스를 활용하여 추적성 분석 결과를 정리한다. 이 과정에서 발견된 이상(anomaly)은 이상보고서를 통해 보고한다.

1.3. SW 구현 검증

SW 구현 검증은 SW 생명 주기 중 구현단계 산출물인 소스 코드에 대한 품질 향상을 목적으로 한다. 소스 코드에 대한 코딩 규칙 준수 및 요구사항 준수 여부에 대한 검증이 수행되어야 한다.

소스 코드 리뷰와 각 수준별 추적성 검증 수행을 위해 SW 확인 및 검증의 대표적인 국제 표준인 IEEE Std. 1012-2004에서 기술하고 있는 방법을 적용한다.

SW 구현 검증 단계에서 수행되는 검증 항목들에 대한 검증 지표는 아래와 같다.

SW 구현 검증 : 소스코드가 설계 문서에 기술된 설계 요구사항을 충족하는지, 코딩 표준을 준수하는지 확인				KC 62619 수행 여부
검증 항목	설명	검증 지표	지표 산정 방법	
소스코드 검증	SW 소스코드가 설계 문서에 기술된 설계 요구사항을 충족하는지 확인	추적성 커버리지	설계 요구사항과 추적된 SW 소스코드 모듈 개수 SW 소스코드 모듈 개수	X
	소스코드가 지정된 코딩 표준을 준수하여 구현되어 있는지 확인	코드 리뷰 체크리스트	적합성 검증 체크리스트 만족/불만족 항목 개수 전체 체크리스트 항목 개수	X

표 4. SW 구현 검증 - 검증 지표

세부 검증 방법 및 절차는 각 항목별 절 내부에 기술되어 있다.

1.3.1. 소스코드 검증

소스코드 검증은 소프트웨어를 실행하지 않은 상태에서 잠재적인 결함을 검출하는 방식으로 이루어지며, 기능 안전에 대한 코딩 규칙 검증 시험, 보안 취약점 점검, 소스코드 메트릭(Code Metrics) 점검이 여기에 포함된다. C 소프트웨어는 검증 시 자동화된 도구를 사용할 수 있다.

1.3.1.1. 기능 안전 검증

기능 안전에 관한 코딩 규정을 적용한 검증은 개발자의 실수를 방지하고 소프트웨어의 예상치 못한 오작동을 줄이는 데 도움이 된다. 향후 자동차, 철도, 원자력 분야로 ESS 사업 범위가 확대될 수 있다. 분야별로 적용하는 코딩 규정이 다르므로 확장하고자 하는 분야에 해당하는 기능 안전 규칙을 적용해야 한다.

기능 안전 검증 진행 시 아래의 코딩 규정을 필요에 따라 선택적으로 적용한다. 도메인과 무관하게 기본적인 기능 안전 검증만 진행하는 경우, MISRA-C를 적용한다. 소스코드 분석 후 위배사항이 없는지 검토하고, 검증 결과는 소스코드 평가 보고서를 통해 보고 한다.

코딩 규정	설명
MISRA-C:2012	C 소프트웨어의 안전성, 신뢰성, 소스코드 품질을 높일 수 있도록 소스코드 검증 규칙을 지원한다.
ISO 26262	차량 내 전기/전자 시스템의 요구사항을 준수하기 위한 지침으로, 개발 활동 및 작업 제품의 안전 관련 측면을 다룬다.
NUREG/CR-6463	NRC(Nuclear Regulatory Commission)가 제공하는 원자력 발전소 SW 구현을 검토하기 위한 지침이다.
IEC 62279	SW 안전 무결성에 대한 지침으로, 각 수준에 필요한 기술과 조치를 포함한다.

표 5. 코딩 규정

범용 C 소프트웨어에는 MISRA-C, 차량 SW를 위한 코딩 규정으로는 ISO 26262, 원자력에는 NUREG/CR-6463, 철도에는 IEC 62279가 있다. 각 코딩 규정의 규칙 목록은 부록 C에서 확인할 수 있다.

1.3.1.2. 보안 취약점 점검

CWE(Common Weakness Enumeration)_C는 보안에 영향을 미칠 수 있는 SW의 취약점 유형을 목록화한 검증 규칙을 제공한다. 제품 출시 전에 소프트웨어 전문가들이 취약점을 점검하고 일반적인 오류를 제거하여 소스코드의 취약성을 경감시키는데 도움을 주는 것이 목적이다.

따라서 개발하는 소프트웨어의 보안성을 높이기 위해서는 소스코드가 이 검증 규칙에서 정의하고 있는 취약점을 포함하고 있는지 점검해야 한다. 결함 검출 시 소스 코드 수정 후 재시험하여 결함 존재 여부를 확인한다.

CWE_C의 규칙 목록은 부록 C에서 확인할 수 있다.

1.3.1.3. 소스코드 메트릭(Metrics)

소스코드 메트릭이란 소프트웨어의 복잡도 감소, 유지보수 용이성 증대 등 소프트웨어의 품질향상을 위한 소스코드의 품질 측정지표를 말한다. 소스코드 메트릭 수치가 과도할 경우, 가독성과 이해도가 떨어지고 결함 확률이 증가하기 때문에 소스코드 메트릭 점검 결과는 코드의 정량화 및 최적화의 기준 만족도로도 활용할 수 있다. 이와 같이 코드의 결함도를 낮추고 응집도를 높이기 위해서는 메트릭의 제한값을 준수해야 하며 검증 결과가 문서화되어야 한다.

소스코드 메트릭 종류는 표 6과 같으며, 소프트웨어에 따라 제한값에는 차이가 있을 수 있다. 또한, 사업에 따라 메트릭 적용 여부를 협의하여 진행할 수 있다.

메트릭 종류	제한값	비 고
Cyclomatic Complexity	20 이하	소프트웨어 흐름에 있는 코드 경로의 수를 계산한 순환 복잡도
Number of Call Levels	6 이하	함수 내 조건문 등의 최대 중첩(Nesting)의 깊이
Number of Function Parameters	8 이하	매개변수의 개수
Number of Calling Functions	8 이하	해당 함수를 호출하는 다른 함수의 개수
Number of Called Functions	10 이하	해당 함수가 호출하는 다른 함수의 개수(같은 함수 호출 시 1로 계산)
Number of Executable Code Lines	200 이하	실행 가능한 코드의 라인 수

표 6. 소스코드 메트릭(Metrics) 종류

1.3.2. 추적성 검증

소스코드가 설계 명세서를 기준으로 하여 완전히, 그리고 정확히 구현했는지 분석하는 과정이다. 정확히 구현됐는지 확인하기 위해서는 코드 리뷰로만 확인하기 보다는 모듈 시험의 결과를 통해 최종 확인하는 것이 바람직하다. 이 과정에서 설계에 명세되지 않은 코드가 포함되었다면 해당 코드의 필요성과 영향성을 고려하여 이상 여부를 결정해야 한다. 반드시 필요한 코드가 아니라면 명세에 없는 코드를 남겨 두는 것은 잠재적 오류의 요인이 될 수도 있으므로 제거하는 것이 좋다.

추적성 분석 결과는 추적성 분석 보고서로 보고한다.

1.4. SW 모듈 시험

1.4.1. 모듈 단위 시험

SW 모듈은 SW가 작동하는 기능의 최소 단위를 의미한다. SW 모듈 시험은 SW 모듈이 명세된 요구사항을 올바르게 수행하는지, 그리고 불필요한 구문이 존재하지는 않는지 확인한다.

SW 모듈 시험은 아래와 같은 절차로 수행된다.

- 1) 테스트 계획 수립 (계획서 작성)

- 2) 분석 및 설계 (절차서 작성)
- 3) 시험 수행 및 보고 (보고서 작성)

SW 모듈 시험의 검증 항목은 코드 커버리지를 통한 모듈 단위 시험과 요구사항 커버리지를 통한 모듈 단위 시험이며 관련 지표는 다음과 같다.

소프트웨어 모듈 시험 : 기능의 최소 단위인 모듈의 데이터 처리 시 발생 가능한 결함 검출				
검증 항목	설명	검증 지표	지표 산정 방법	KC 62619
모듈 단위 시험	소프트웨어 단위 모듈이 기능 요구사항을 수행하는 중 구문 또는 분기 등 도달하지 않는 부분이 있는지 테스트 케이스를 생성하고 확인한다.	코드 커버리지 (구문, 분기, MC/DC)	시험에서 수행된 코드 수 전체 코드 수	X
	모듈 설계서에 명세된 값을 기반으로 테스트 케이스를 생성하고 소프트웨어 단위 모듈이 명세된 요구사항을 충족하는지 동작시켜 확인한다.	요구사항 커버리지	시험에서 충족된 요구사항 개수 SW 모듈 설계 요구사항 개수	X

표 7. SW 모듈 시험 지표

1.4.1.1. SW 모듈 시험 계획서 작성

SW 설계 기술서에 기술된 각 모듈들이 시험 대상이 되며 SW 모듈이 정확하게 요구사항을 이행하고 있는지 검증하기 위해 모듈 시험을 계획해야 한다.

모듈 시험 계획서 작성에 앞서 필요한 문서의 목록은 아래와 같으며, 해당 문서들이 준비되면 계획서 작성을 시작할 수 있다.

- SRS(Software Requirement Specification)
- SDD(Software Design Description)
- SVVP(Software Verification & Validation Plan)

SW 모듈 시험 계획서에서 기술되어야 하는 내용은 다음과 같다.

- 1) 시험 설계, 사례, 절차 및 결과에 대한 설계 요구사항의 추적
- 2) 시험 업무 및 결과의 문서화
- 3) 시험 계획이 다음의 기준을 만족하는지 검증
 - 가) 설계요건 준수
 - 나) 타이밍, 크기 및 정확성 평가
 - 다) 경계 및 인터페이스 성능, 스트레스 및 오류 조건 하에서의 성능
 - 라) 요구사항 시험 범위, SW 신뢰성 및 유지보수

1.4.1.2. SW 모듈 시험 절차서 작성

SW 설계 기술서에 기술된 각 모듈들이 시험 대상이 되며 SW 모듈이 정확하게 요구사항을 이행하고 있는지 검증하기 위해 모듈 시험 절차서를 설계해야 한다.

모듈 시험 절차서 작성에 앞서 필요한 문서의 목록은 아래와 같으며, 해당 문서들이 준비된 뒤에 절차서 작성을 시작할 수 있다.

- SRS(Software Requirement Specification)

- SDD(Software Design Description)

SW 모듈 시험 절차서에서 기술되어야 하는 내용은 다음과 같다.

- 1) SW 모듈 시험 계획서와의 추적
- 2) SW 모듈 시험에 대한 상세 절차
- 3) SW 모듈 시험에 대한 시험 사례
- 4) SW 모듈 시험 환경 및 예외 사항

1.4.1.3. SW 모듈 시험 수행 및 보고

SW 모듈이 정확하게 구성요소 요건을 이행하고 있는지 검증하기 위해 시행되어야 한다.

모듈 시험 수행 및 보고서 작성에 앞서 필요한 문서의 목록은 아래와 같으며, 해당 문서들이 준비된 뒤에 시험 수행 및 보고서 작성을 시작할 수 있다.

- 소스 코드
- 실행가능 코드
- SDD(Software Design Description)
- 시험 계획
- 시험 절차

SW 모듈 시험을 시행하고 보고서에서 기술되어야 하는 내용은 다음과 같다.

- 1) V&V 모듈 시험 시행
- 2) 시험 계획 요건에 따라 결과를 문서화
- 3) 시험 결과가 다음의 기준을 만족하는지 검증
 - 가) V&V 시험 허용기준 만족 여부
 - 나) 설계내용 이행의 정확성
 - 다) 계획문서에 있는 시험 추적성에 의해 수립된 기준을 시험한 결과의 추적성
- 4) 실제 시험 결과와 예상 결과와의 차이점을 문서화

1.5. SW 통합 시험

1.5.1. 모듈 통합 시험

SW 통합 시험은 아래와 같은 절차로 수행된다.

- 1) 테스트 계획 수립 (계획서 작성)
- 2) 분석 및 설계 (절차서 작성)
- 3) 시험 수행 및 보고 (보고서 작성)

SW 통합 시험의 검증 항목은 함수 호출 커버리지를 통한 인터페이스 통합 시험과 아키텍처 통합 시험이며 관련 지표는 다음과 같다.

소프트웨어 통합 시험 : 모듈의 통합이 조건과 설계를 이행하고 있는지 검증				
검증 항목	설명	검증 지표	지표 산정 방법	KC 62619
인터페이스 통합 시험	개발한 함수를 타 함수가 호출(사용) 가능한지, 호출(사용)하고 있는지 확인하는 시험	함수 커버리지	$\frac{\text{호출 성공한 함수 수}}{\text{전체 함수 수}}$	X
아키텍처 통합 시험	아키텍처 기준으로 모듈을 통합하고 최상위 함수에서 말단 함수까지 호출/사용 가능한지 확인하는 시험	호출 커버리지	$\frac{\text{달성한 함수 호출 수}}{\text{전체 함수 호출 수}}$	X

표 8. SW 통합 시험 지표

지표 산정 시 라이브러리는 제외한다. 기타 세부 검증 방법 및 절차는 각 항목별 절 내부에 기술되어 있다.

1.5.1.1. SW 통합 시험 계획서 작성

SW 설계 기술서에 기술된 각 모듈들이 통합되어 감에 따라 정확하게 모듈 조건과 설계를 이행하고 있는지 검증하기 위해 통합 시험을 계획해야 한다.

통합 시험 계획서 작성에 앞서 필요한 문서의 목록은 아래와 같으며, 해당 문서들이 준비되면 계획서 작성을 시작할 수 있다.

- SRS(Software Requirement Specification)
- SDD(Software Design Description)
- SVVP(Software Verification & Validation Plan)

SW 통합 시험 계획서에서 기술되어야 하는 내용은 다음과 같다.

- 1) 시험 설계, 사례, 절차 및 결과에 대한 설계 요구사항의 추적
- 2) 시험 업무 및 결과의 문서화
- 3) 시험 계획이 다음의 기준을 만족하는지 검증
 - 가) 통합 수준 별 기능 요구사항 준수
 - 나) 타이밍, 크기 및 정확성 평가
 - 다) 스트레스 및 오류 조건 하에서의 성능
 - 라) 요구사항 시험 범위 SW 신뢰성 측정

1.5.1.2. SW 통합 시험 절차서 작성

SW 설계 기술서에 기술된 각 모듈들이 통합되어 감에 따라 정확하게 모듈 요건과 설계를 이행하고 있는지 검증하기 위해 통합 시험 절차서를 설계해야 한다.

통합 시험 절차서 작성에 앞서 필요한 문서의 목록은 아래와 같으며, 해당 문서들이 준비된 뒤에 절차서 작성을 시작할 수 있다.

- SRS(Software Requirement Specification)
- SDD(Software Design Description)

SW 통합 시험 절차서에 기술되어야 하는 내용은 다음과 같다.

- 1) SW 통합 시험 계획서와의 추적
- 2) SW 통합 시험에 대한 상세 절차
- 3) SW 통합 시험에 대한 시험 사례
- 4) SW 통합 시험 환경 및 예외 사항

1.5.1.3. SW 통합 시험 수행 및 보고

SW 설계 기술서에 기술된 각 모듈들이 통합되어 감에 따라 정확하게 모듈 요건과 설계를 이행하고 있는지 검증하기 위해 통합 시험을 시행해야 한다.

통합 시험 수행 및 보고서 작성에 앞서 필요한 문서의 목록은 아래와 같으며, 해당 문서들이 준비된 뒤에 시험 수행 및 보고서 작성을 시작할 수 있다.

- 소스 코드
- 실행가능 코드
- 시험 계획
- 시험 절차

SW 통합 시험을 시행하고 보고서에서 기술되어야 하는 내용은 다음과 같다.

- 1) V&V 통합요소 시험 시행
- 2) 시험 계획의 요건에 따라 결과를 문서화
- 3) 시험 결과가 다음의 기준을 만족하는지 검증
 - 가) SW 구성요소 통합의 정확성
 - 나) 시험 추적성에 의해 수립된 시험기준에 대한 시험 결과의 추적성
 - 다) V&V 시험 허용기준 만족 여부
- 4) 실제 시험 결과와 예상 결과와의 차이점을 문서화

1.6. 시스템 통합 시험

1.6.1. SW 시스템 통합 시험

시스템 통합 시험은 BMS SW와 HW가 결합할 시 명세된 기능 및 비기능 요구사항을 충족하는지 확인하고 스트레스 조건에서도 요구사항을 어느 정도 동작할 수 있는지 확인한다. 해당 시험은 KC 62619에서도 오류 주입 시험을 통하여 일부 수행하고 있다. 해당 과정에서 검증 항목으로 시스템 기능 시험, 시스템 비기능 시험 및 스트레스 조건에서의 시스템 강건성 시험을 제시한다.

SW 시스템 통합 시험은 아래와 같은 절차로 수행된다.

- 1) 테스트 계획 수립 (계획서 작성)
- 2) 분석 및 설계 (절차서 작성)
- 3) 시험 수행 및 보고 (보고서 작성)

SW 시스템 통합 시험 검증 지표는 다음과 같다.

시스템 통합 시험 : BMS 소프트웨어와 하드웨어 간 상호작용시 발생 가능한 오류 검출				
검증항목	설명	검증 지표	지표 산정 방법	KC 62619
시스템 기능 시험	시스템 또는 SW 요구사항의 기능 명세를 충족하는지 시험한다.	기능 요구사항 커버리지	$\frac{\text{시험에서 충족된 기능 요구사항 개수}}{\text{기능 요구사항 개수}}$	\triangle 8.2에서 안전기능 기능시험 수행
시스템 비기능 시험	시스템의 자원 사용량, 응답 시간 등의 비기능 요구사항 명세를 충족하는지 시험한다.	비기능 요구사항 커버리지	$\frac{\text{시험에서 충족된 비기능 요구사항 개수}}{\text{비기능 요구사항 개수}}$	X
시스템 강건성 시험	결함 주입, 부하 등의 스트레스 조건에서 시스템이 정지 없이 동작 가능한지 시험한다.	강건성	$\frac{\text{시스템 무정지 TC 개수}}{\text{스트레스 조건 TC 개수}}$	\triangle 부속서 D.5에서 안전기능 기능시험 수행

표 9. 시스템 통합 시험 검증 지표

세부 검증 방법 및 절차는 각 항목별 절 내부에 기술되어 있다.

1.6.1.1. SW 시스템 통합 시험 계획서 작성

완전한 SW 최종제품으로서 모든 요건의 준수함을 검증하기 위한 시스템 통합 시험을 계획해야 한다.

통합 시험 계획서 작성에 앞서 필요한 문서의 목록은 아래와 같으며, 해당 문서들이 준비되면 계획서 작성을 시작할 수 있다.

- 개념문서(필요시)
- SRS(Software Requirement Specification)
- SVVP(Software Verification & Validation Plan)

SW 시스템 통합 시험 계획서에서 기술되어야 하는 내용은 다음과 같다.

- 1) 시험 설계, 사례, 절차, 결과와 이에 대한 요건 추적
- 2) 시스템 통합 환경에서의 완전한 SW 최종제품으로서 모든 요건의 준수성
- 3) 사용자 문서의 적합성
- 4) 스트레스 및 오류 조건 하에서의 성능
- 5) 시스템 통합 요건의 시험 범위

1.6.1.2. SW 시스템 통합 시험 절차서 작성

완전한 SW 최종제품으로서 모든 요건의 준수함을 검증하기 위한 시스템 통합 시험 절차를 설계해야 한다.

시스템 통합 시험 절차서 작성에 앞서 필요한 문서의 목록은 아래와 같으며, 해당 문서들

이 준비된 뒤에 절차서 작성을 시작할 수 있다.

- SRS(Software Requirement Specification)
- SDD(Software Design Description)

SW 시스템 통합 시험 절차서에 기술되어야 하는 내용은 다음과 같다.

- 1) SW 시스템 통합 시험 계획서와의 추적
- 2) SW 시스템 통합 시험에 대한 상세 절차
- 3) SW 시스템 통합 시험에 대한 시험 사례
- 4) SW 시스템 통합 시험 환경 및 예외 사항

1.6.1.3. SW 시스템 통합 시험 수행 및 보고

완전한 SW 최종제품으로서 모든 요건의 준수함을 검증하기 위한 시스템 통합 시험을 시행해야 한다.

시스템 통합 시험 수행 및 보고서 작성에 앞서 필요한 문서의 목록은 아래와 같으며, 해당 문서들이 준비된 뒤에 시험 수행 및 보고서 작성을 시작할 수 있다.

- 소스 코드
- 실행가능 코드
- 시험 계획
- 시험 절차

SW 시스템 통합 시험을 시행하고 보고서에서 기술되어야 하는 내용은 다음과 같다.

- 1) V&V 시스템 통합 시험 시행
- 2) 시험 계획의 요건에 따라 결과를 문서화
- 3) 시험 결과가 다음의 기준을 만족하는지 검증
 - 가) 시스템 통합 요건 만족 여부
 - 나) 시험 추적성에 의해 수립된 시험기준에 대한 시험 결과의 추적성
 - 다) V&V 시험 허용 기준 만족 여부
- 4) 실제 시험 결과와 예상 결과와의 차이점을 문서화

2. SW V&V 보고

2.1. V&V 보고요건

V&V 보고는 SW 생명 주기 내내 발생하며, 수행된 각각의 V&V 업무에 대해 각각의 요구 출력자료를 생성해야 한다. 또한 이들 공정, 활동 및 업무를 관리하고 수행하며 기타 관련된 사업적 측면에서 조정하는데 필요한 정보와 시설을 규정하여야 한다. V&V 보고서의 형태 및 분류는 사용자에게 의해 정의될 수 있으며 대략적인 구성은 아래와 같다.

(1) V&V 업무 보고서 : V&V 업무 결과 및 상태에 관한 내용이며 아래와 같은 업무들에 관한 보고서이다.

- 1) 검토 결과
- 2) 추적성 검증
- 3) SW 요구사항 검증
- 4) SW 상세설계 검증
- 5) 소스 코드 및 소스 코드 문서 검증
- 6) 시험 결과

V&V 활동 요약보고서를 V&V 활동 보고서로 작성하는 경우, 위 V&V 업무 보고서들은 V&V 활동 보고서의 부록으로 작성할 수 있다.

(2) V&V 활동 요약보고서 : 아래 V&V 생명 주기 활동에 대하여 수행된 V&V 업무 결과를 요약해야 한다.

- 1) SW 설계 검증
- 2) SW 구현 검증
- 3) SW 시험

V&V 활동 요약보고서의 내용은 아래와 같이 작성할 수 있으며, 예시의 목차는 일반적인 사항들로 구성되어 있으며 필요한 경우 목차를 조정할 수 있다.

V&V 활동 요약보고서 목차	
1.0 개요	
1.1 목적	
1.2 범위	
1.3 참조 문서	
2.0 평가 개요	
3.0 이상 보고	
4.0 평가 결과	
5.0 결론	
6.0 일반 사항	
6.1 용어 정의	
6.2 약어	

(3) V&V 이상보고서 : V&V 결과로 발견된 모든 이상이 문서화 되어야 하며 아래 사항들을 기준으로 작성한다.

- 1) 이상의 등록번호 및 제목
- 2) 문서 버전
- 3) 이상 대상 기기/계통 명칭
- 4) 이상 발견 날짜
- 5) 이상이 발견된 문서의 정보
- 6) 식별된 이상의 현상과 상태
- 7) 조치방안
- 8) 조치 결과

(4) V&V 최종보고서 : 설치나 검사 활동 완료 시점 또는 V&V 업무 종료 시점에 발행되어야 한다. V&V 최종보고서의 내용은 아래와 같이 작성할 수 있으며, 예시의 목차는 일반적인 사항들로 구성되어 있으며 필요한 경우 목차를 조정할 수 있다.

V&V 최종보고서 목차	
1.0	목적
2.0	범위
3.0	참고문서
4.0	용어집
5.0	V&V 활동
5.1.	검증 계획 작성
5.2.	SW 설계 검증
5.3.	SW 구현 검증
5.4.	SW 모듈 시험
5.5.	SW 통합 시험
5.6.	시스템 통합 시험
6.0	결론

(5) 선택적 V&V 보고서 : V&V 보고서는 필요시 특수연구보고서, 기타 업무보고서와 같은 선택보고서를 포함할 수 있다. SW 생명 주기 기간 중 수행된 특수한 V&V 연구에 대하여 특수연구보고서에 문서화해야 하며, SVVP에 정의되지는 않았지만, 실제 수행된 업무의 결과를 보고서에 문서화해야 한다. 기타 업무보고서에는 품질보증 결과, 최종사용자 시험 결과, 안전성 평가 보고서 또는 형상 및 데이터 관리 결과를 포함할 수 있다. 보고서의 제목은 주제의 성격에 따라 달라질 수 있다.

업무보고서, V&V 활동 요약보고서와 이상보고서는 각 SW 성과물과 공정의 기술 품질에 관한 SW 개발공정으로서의 피드백용으로 제공된다.

3. V&V 관리 절차

3.1. 관리공정

관리공정은 아래와 같은 일반적 활동 및 업무로 이루어진다.

- 공정실행 계획 준비
- 계획이행의 개시
- 계획실행의 감시
- 계획실행 중에 발견된 문제점 분석
- 공정 진도 보고
- 제품의 요건충족 보증
- 평가 결과 평가
- 업무 종결 여부 판단
- 결과의 완결성 점검
- 효율성 및 유효성에 대한 공정 점검
- 사업 품질검토
- 사업위험도 검토
- 사업 수단 검토

3.1.1. V&V 관리 활동

V&V 관리 활동은 모든 SW 생명 주기 공정 및 활동에 대해 수행되며, 이를 통해 V&V 출력물을 감시하고 평가한다. 관련 활동은 아래와 같다.

- V&V 노력의 지속적 검토
- 갱신된 사업 일정과 개발상황을 기준으로 하여 필요에 따라 SVVP 개정
- V&V 결과를 토대로 개발 분야와 품질보증, 형상 관리 등과 같은 다른 지원 공정에 대한 조정
- 검토 및 감사 수행
- V&V 수행 시 공정향상 기회 식별

V&V 관리에서는 계통 및 SW의 변경된 내용을 평가하고, 이들 변경으로 영향을 받는 SW 요건을 식별하며, 변경을 처리하는 V&V 업무를 계획한다. 각각 제안된 변경에 대해 V&V 관리 활동은 SW에 새로운 위험 요소나 위험도의 이입 여부를 평가하고 지정된 SW 무결성 등급이 변경으로 영향을 받는 지를 확인한다. 만일 SW 무결성 등급이나 위험 요소 및 위험도가 변경될 때는 새로운 V&V 업무의 추가나 기존의 V&V 업무 범위와 강도를 증가시키는 것으로 V&V 업무계획을 개정한다. 기준선의 변경은 점진적 SW 개발공정에서 SW 출시에 배정되는 변경에서 기인한다(예 : 계획된 기준선 버전).

V&V 측정과 다른 정성적 및 정량적 척도를 이용하여 이 V&V 활동에서 개발자와 취득자가 적시에 인지하여 해결할 수 있도록 프로그램 추이 데이터 및 가능한 위험도 문제를 제시한다. 주요 프로그램 공정 시점(예 : 요건검토, 설계검토, 시험 준비 완료)마다 V&V 관리에서는 V&V 결과를 토대로 SW 개발 활동을 다음 과정으로의 진행 여부에 대한 확고한 증거를 수립한다. 필요한 경우에는 언제든지 V&V 관리를 통해 SW 프로그램의 변경에 따른 V&V 업무의 재수행 여부를 결정한다.

V&V 업무에서는 선택된 SW 무결성 등급에 따라 다음에 열거한 최소 V&V 업무를 수행하여야 한다.

- (1) 업무 : SVVP 작성
- (2) 업무 : 기준선(baseline) 변경 평가
- (3) 업무 : V&V 관리 검토
- (4) 업무 : 관리 및 기술 검토 지원
- (5) 업무 : 조직 및 지원 공정과의 인터페이스
- (6) 업무 : V&V 수행 중 공정 향상 기회 확인

4. 결론

대한민국 정부는 지구온난화 기후 위기 속에서 탄소 발생을 줄이고 단계적으로 석탄 발전 가동을 중단하여 2030년까지 전체 발전량의 20%를 신재생에너지로 공급하는 것을 목표로 하고 있다. 이런 환경에서 ESS는 신재생에너지 발전으로 생산한 전기를 이차전지에 저장했다가 필요시 공급하여 주파수 조정, 신재생에너지 연계 발전, 전력 피크 억제 효과, 전력 수급 위기 대응 등 많은 이점을 가지고 있다. 그러나 2017년부터 발생한 ESS 화재 사고 이후 늘어난 규제로 인해 국내 ESS 시장은 정체되어있는 상황이다.

조사위는 ESS 화재 사고에 대해 배터리 보호시스템의 미흡, ESS 통합 제어체계의 미흡 등을 원인으로 밝혔다. 이에 대해, ESS의 전력 흐름을 통합적으로 제어하고 관리하는 BMS는 종합적인 안전 강화 대책이 될 수 있다.

즉, BMS는 SW 기능 안전과 신뢰성이 보장되어야 하며, 이를 위해서는 SW 안전 검증이 필수적이다. 이러한 SW 안전 검증을 통해 BMS의 신뢰성이 확보되면 ESS 시스템의 오류로 인한 사고를 미리 감지/방지할 수 있게 되고, ESS의 전체적인 안전성을 높일 수 있게 된다.

점진적으로 ESS의 안전 검증 체계가 자리를 잡아 ESS의 안전성을 최고조로 끌어올릴 수 있게 되면, 국내 ESS 기업이 세계 ESS 시장의 선두에 설 수 있으리라 기대한다.

부록 A. 체크리스트

소프트웨어 V&V 체크리스트 - IEEE 1471

번호	장절	점검항목
1	1.1 Scope	<p>범위는 다음과 같이 사용되는 아키텍처 설명을 포함하고 있는가?</p> <ul style="list-style-type: none"> a) 시스템 및 그 진화의 표현 b) 시스템 이해 관계자들 간의 의사소통 c) 일관된 방식으로 아키텍처 평가 및 비교 d) 시스템 개발의 활동 계획, 관리 및 실행 e) 시스템의 지속적인 특성 및 지원 원칙 표현으로서 수용 가능한 변경을 안내 f) 아키텍처 설명과의 시스템 구현의 준수 여부 g) 소프트웨어 집약적인 시스템 아키텍처 지식 기록에 대한 기여
2	1.3 Intended users	<p>주요 사용자 그룹은 시스템 개발과 진화에 관련된 이해 관계자들로서 다음과 같은 사람들을 포함하고 있는가?</p> <ul style="list-style-type: none"> - 시스템을 사용, 소유 및 획득하는 사람들 (사용자, 운영자 및 획득자 또는 클라이언트) - 아키텍처를 개발하고 설명하며 문서화하는 사람들 (아키텍트) - 시스템을 개발, 제공 및 유지보수하는 사람들 (아키텍트, 디자이너, 프로그래머, 유지보수자, 테스터, 도메인 엔지니어, 품질 보증 직원, 구성 관리 직원, 공급 업체 및 프로젝트 관리자 또는 개발자들) - 시스템 및 그 개발을 감독하고 평가하는 사람들 (최고 정보 담당자, 감사인 및 독립 평가자들)
3		<p>보조 사용자 그룹은 여러 시스템 개발을 포괄하는 기업 전반의 인프라 활동에 참여하는 사람들로서, 다음의 사람들을 포함하고 있는가?</p> <ul style="list-style-type: none"> - 방법론자 - 프로세스 및 프로세스 개선 엔지니어 - 연구자 - 표준 제작자 - 도구 개발자 및 트레이너
4	1.4 Conformance to this recommended practice	<p>아키텍처 설명은 5절에 나열된 요구사항을 충족하고 있는가?</p>

번호	장절	점검항목
5	4.1 Architectural description in context	시스템은 하나 이상의 이해 관계자를 가지고 있으며, 시스템의 우려사항은 성능, 신뢰성, 보안, 배포성 및 발전 가능성과 같은 시스템 고려사항을 포함되어 있는가?
6		개념적인 프레임워크에서 아키텍처 설명은 하나 이상의 구성 요소인 (아키텍처) 뷰로 구성되어 있는가?
7		뷰 포인트는 뷰를 기술하는 데 사용할 언어(표기법, 모델 또는 제품 유형 등)와 뷰의 표현에 적용할 모델링 방법 또는 분석 기법을 결정하는가?
8		아키텍처 설명은 하나 이상의 관점을 선택하여 사용하고 있는가?
9		관점을 선택하는 것은 일반적으로 AD가 주소되는 이해 관계자들과 그들의 관심사를 고려하여 이루어지는가?
10		각각의 아키텍처 모델은 연관된 아키텍처 관점에 의해 정해진 방법을 사용하여 개발되는가?
11	4.2 Stakeholders and their roles	이해 관계자들은 아키텍처 설명의 작성과 사용과 관련하여 다양한 역할을 가지고 있는가?
12		이해 관계자들에는 클라이언트, 사용자, 아키텍트, 개발자, 평가자 등이 포함되어 있는가?
13		이해관계자들 중 두 가지 주요 역할은 수요자(또는 클라이언트)와 아키텍트인가?
14		아키텍트는 시스템의 이해 관계자들을 위해 아키텍처를 아키텍처 설명의 형태로 기록하고 있는가?
15	4.3 Architectural activities in the life cycle	아키텍처 설계는 시스템이 처음으로 개념화되어 사용이 종료될 때까지의 개발, 운영 및 유지 보수에 기여되고 있는가?
16		아키텍처 설계는 만족스럽고 실행 가능한 시스템 개념을 개발하는 데 관여하고 있는가?
17		아키텍처 설계는 개발을 통해 해당 시스템 개념의 무결성을 유지하고 있는가?
18		아키텍처 설계는 구축된 시스템을 사용하기 위해 인증하고 있는가?
19		아키텍처 설계는 운영 및 진화 단계에서 해당 시스템 개념을 보장하는 데 관여하고 있는가?

번호	장절	점검항목
20		시스템 아키텍처 개발 이후에는 일반적으로 자세한 시스템 엔지니어링 활동 (자세한 요구사항 정의 및 인터페이스 명세, 주요 하위 시스템의 아키텍처 설계)과 같은 작업이 진행었는가?
21		사용자들은 권장 사항을 사용할 때, 따로 수명 주기 모델을 선택하고 있는가?
22	4.3.1 Scenario: architecture of single systems	다음에 해당 되는 경우 아키텍처 설계는 사용자-발주자의 시스템 비전, 시스템 목표 및 이해관계자의 요구사항과 제약사항에 대응해 진행되는가? - 새로운 시스템 구축의 경우 - 사용자와 발주자가 동일한 경우
23	4.3.2 Scenario: iterative architecture for evolutionary systems	초기 아키텍처는 현재와 예상되는 사용자 요구사항과 현재 및 추정된 기술 능력에 대응하여 준비되는가?
24		초기 시스템은 이러한 아키텍처를 기반으로 전달되고 있는가?
25		아키텍처가 새로운 이해관계자들의 요구사항에 대응하여 진화함에 따라, 해당 개발 당시의 현재 아키텍처를 기반으로 시스템이 전달되고 있는가?
26		시스템과 아키텍처에 포함된 이해관계자들의 집합도 아키텍처가 진화함에 따라 변경되고 있는가?
27		진화형 시스템의 아키텍처 설명서는 시스템 개발과 평가에 사용되며, 그 사용과 개발은 동시에 진행되고 있는가?
28		각 아키텍처 반복마다 이해관계자의 요구사항을 재평가 하고있는가?
29		아키텍처 설명서는 각 시스템이 개발을 진행함에 따라 이를 안내하는 데 사용되고 있는가?
30		아키텍처 설명서는 전달 시에 적합한 아키텍처 설명서 버전을 사용하여 각 시스템을 평가하는 데 사용되고 있는가?
31		진화하는 시스템에 대한 아키텍처 설명서는 일반적으로 반복적인 버전 변경의 패턴 또는 리듬으로 개발되었는가?
32		일반적으로 아키텍처 개발은 개발자에 의해 시스템 시퀀스를 개발하는 전반적인 활동의 일부로 진행되고 있는가?
33	4.3.3	이 권장 사항을 사용하여, 기존 시스템의 아키텍처 기술을 먼저

번호	장절	점검항목
	Scenario:	작성하였는가?
34	architecture of existing systems	이 아키텍처기술을 사용하여 새로운 시스템을 개발하거나 유지보수 또는 진화 활동을 지원하거나, 위의 시나리오들 (4.3.1 및 4.3.2 참조)과 같이 진화 아키텍처를 수립하는 데 활용되었는가?
35		아키텍처 기술서의 품질은 일반적으로 기술서의 이해 가능성, 일관성, 완전성 및 분석 가능성을 포함하였는가?
36	4.3.4 Scenario:	아키텍처 기술서로부터 얻어지는 시스템의 품질을 예측하는 것은 실행 가능성, 효율성, 그리고 신뢰성과 같은 품질을 포함하였는가?
37	architectural evaluation	아키텍처가 아키텍처 기술서에 준수하는지 평가하기 위해, 구현으로부터 아키텍처 기술서를 역공학하는 프로세스(4.3.3절 참조)를 적용하였는가?
38	4.4 Uses of architectural descriptions	<p>아키텍처 설명은 수명주기동안 이해관계자들에 의해 다음과 같은 다양한 용도로 활용되고 있는가? (이에 국한되지는 않음)</p> <p>a) 대체 아키텍처 분석</p> <p>b) 레거시 아키텍처에서 새로운 아키텍처로의 전환을 위한 비즈니스 계획 수립</p> <p>c) 시스템 개발, 생산, 실무, 운영 및 유지보수와 관련된 조직 간 커뮤니케이션</p> <p>d) 계약 협상의 일환으로 수요기관과 개발자들 간의 커뮤니케이션</p> <p>e) 아키텍처와 구현 사이의 준수 여부를 인증하는 기준</p> <p>f) 개발 및 유지보수 문서 작성, 재사용 저장소 및 교육 자료 포함</p> <p>g) 후속 시스템 설계 및 개발 활동에 대한 입력</p> <p>h) 시스템 생성 및 분석 도구에 대한 입력</p> <p>i) 운영 및 인프라 지원; 구성 관리 및 시스템, 하위 시스템 및 구성 요소의 재설계 및 유지보수</p> <p>j) 계획 및 예산 지원</p> <p>k) 획득 문서 작성(예: 제안 요청서 및 작업 내용서)</p> <p>l) 수명 주기 동안 시스템의 검토, 분석 및 평가</p> <p>m) 공통 기능 세트를 공유하는 일련의 시스템에 대한 명세서(예: 제품 라인)</p>
39	5. Architectural description	<p>아키텍처 설명의 수행방법을 정의하며, 이 요건을 충족하고 있는가?</p> <p>a) 아키텍처 설명(AD) 식별, 버전 및 개요 정보 (5.1 참조)</p>

번호	장절	점검항목
	practices	b) 아키텍처에 관련된 시스템 이해 관계자들과 그들의 관심사의 식별 (5.2 참조) c) 아키텍처 표현을 조직화하는 데 선택된 각 관점(viewpoint)의 명세 및 그 선택 이유 (5.3 참조) d) 하나 이상의 아키텍처 뷰 (5.4 참조) e) 아키텍처 설명의 필수 요소들 간에 알려진 모든 모순들의 기록 (5.5 참조) f) 아키텍처 선택에 대한 근거 (5.6 참조)
40	5.1 Architectural documentation	아키텍처 설명(AD)은 전체적으로 다음과 같은 정보를 포함하고 있는가? a) 발행일 및 상태 b) 발행 기관 c) 변경 이력 d) 요약 e) 범위 f) 맥락 g) 용어집 h) 참고 문헌
41		상세한 양식과 내용은 IEEE/EIA Std 12207.0-1996의 요구 사항을 만족시킬 수 있도록 선택되었는가?
42	5.2 Identification of stakeholders and concerns	아키텍처(AD)설명은 아키텍트가 시스템의 아키텍처 개념을 구성하는 데 고려한 다음을 식별하고 있는가? a) 시스템 사용자 b) 시스템 획득자 c) 시스템 개발자 d) 시스템 유지보수자 또한 아키텍트 설명(AD)설명은 아키텍처가 시스템의 아키텍처 개념을 구성하는 데 고려한 관심사들을 식별
43		다음의 관심사들의 식별하고 있는가? - 시스템의 목적 또는 임무 - 시스템이 임무를 수행하는 데 적합한지 여부 - 시스템 구축의 실현 가능성 - 시스템 개발 및 운영으로 인한 위험 (사용자, 획득자 및 시스템 개발자들에게) - 시스템의 유지보수, 배포 가능성 및 발전 가능성 - "임무(mission)", "적합성(appropriateness)", 그리고 "실현 가

번호	장절	점검항목
		능성(feasibility)"이라는 용어의 구체적인 의미는 시스템마다 다를 수 있다.
44		아키텍처 설명(AD)은 관심사들을 특정 시스템의 문맥에 맞추어 구체적으로 명시하고 있는가?
45	5.3 Selection of architectural viewpoints	아키텍처 설명(AD)은 사용하기 위해 다음과 같은 뷰 포인트들을 식별해야 하고 있는가? a) 뷰 포인트의 이름(viewpoint name) b) 뷰 포인트를 통해 해결하려는 이해 관계자(stakeholders) c) 뷰 포인트를 통해 해결하려는 관심사(concerns) d) 뷰 포인트에 기반한 뷰(view)를 구성하는 데 사용되는 언어, 모델링 기법 또는 분석 방법 e) 라이브러리 뷰 포인트(library viewpoint)의 경우 출처(사용 기관에 따라 작성자, 날짜 또는 다른 문서에 대한 참조 등)
46		아키텍처 설명(AD)은 각 뷰 포인트(viewpoint)의 선택에 대한 근거를 포함하고 있는가?
47		근거는 5.2절에서 요구된 이해 관계자들과 관심사들이 이 절에서 선택된 뷰 포인트들에 의해 얼마나 다루어지는지 정의하고 있는가?
48		아키텍처 설명(AD)에서 5.2절에 따라 식별된 각 이해 관계자와 각 관심사는 5.3절에 따라 적어도 하나의 뷰 포인트(viewpoint)를 다루고 있는가?
49	5.4 Architectural views	아키텍처 설명(AD)은 하나 이상의 아키텍처 뷰(architectural view)를 포함하고 있는가?
50		아키텍처 설명(AD)의 각 뷰(view)는 5.3절에 따라 선택된 뷰 포인트(viewpoint)와 정확히 대응하고 있는가?
51		아키텍처 설명(AD)의 각 뷰(view)는 해당하는 뷰 포인트(viewpoint)의 명세에 따라 준수되고 있는가?
52		각 뷰(view)는 다음을 포함하고 있는가? a) 사용 기관에서 정의한 식별자와 기타 시작 정보 b) 관련된 뷰 포인트(viewpoint)의 언어, 방법 및 모델링 또는 분석 기법을 사용하여 구성된 시스템의 표현 c) 구성 정보(configuration information)는 사용 기관에서 정의한 대로 포함되어야 한다. d) 아키텍처 뷰(architectural view)는 하나 이상의 아키텍처 모

번호	장절	점검항목
		텔(architectural model)로 구성될 수 있다.
53	5.5 Consistency among architectural views	아키텍처 설명(AD)는 아키텍처 뷰들 간에 알려진 모순들을 기록하고 있는가?
54		아키텍처 설명(AD)는 모든 아키텍처 뷰들 간의 일관성 분석을 포함하고 있는가?
55	5.6 Architectural rationale	아키텍처 설명(AD)은 선택된 아키텍처 개념에 대한 근거를 포함하고 있는가?
56		아키텍처 설명(AD)은 대안적인 아키텍처 개념의 고려와 선택의 근거에 대한 증거를 제공하고 있는가?
57		아키텍처 설명(AD)의 개발 이전에 존재하는 시스템을 설명할 때는 기존 시스템 아키텍처에 대한 근거를 제시하고 있는가?
58	5.7 Example use (informative)	이 권장 사례에 준수함을 주장하기 위해, 아키텍트는 5.1절 부터 5.6절까지의 요구사항을 만족시키는 아키텍처 설명을 작성하였는가?
59		5.1 절에서는 아키텍처 설명(AD)에 기본적인 문서화 정보를 포함하였는가?
60		5.2에서는 아키텍처 설명(AD)가 아키텍처적으로 관련 있는 이해관계자들과 관심사들을 식별하였는가?
61		5.3 절에서는 뷰 포인트(viewpoint)들의 집합을 선택하고 정의하고, 그것을 다루는 범위를 분석하며, 그들의 선택에 대한 근거를 제시하고 있는가?
62		아키텍트는 해당 관점 아래 시스템 동작을 설명하는 데 사용될 언어 또는 모델들을 정의하고 있는가?
63		사용되는 언어나 모델링 기법에는 제한이 없지만, 5.3절에 따라 명시적으로 지정하고 있는가?
64		5.4 절에서는 결과적인 아키텍처를 하나 이상의 뷰(view)를 통해 표현하고 있는가?
65		5.5 절에서는 뷰들 간 또는 뷰들 내에 알려진 모순들 (예: 제공된 행위적 뷰와 구조적 뷰 사이의 모순)이 아키텍처 설명(AD)에 명시하고 있는가?
66		5.6절에 따라 아키텍처 설명(AD)은 아키텍트가 내린 아키텍처적

번호	장절	점검항목
		결정과 선택들에 대한 근거를 포함하고 있는가?

표 10. 소프트웨어 V&V 체크리스트 - IEEE 1471

소프트웨어 V&V 체크리스트 - IEEE 830

번호	장절	점검항목
1	5.1.1 Purpose (1.1 of the SRS)	SRS 목적을 설명하고 있는가?
2		SRS는 의도된 청중을 지정하고 있는가?
3	5.1.2 Scope (1.2 of the SRS)	생산될 소프트웨어 제품의 이름을 식별하고 있는가?
4		소프트웨어 제품이 무엇을 할지, 그리고 필요한 경우 무엇을 하지 않을지 설명하고 있는가?
5		명시되는 소프트웨어의 적용을 설명하고, 관련 이점, 목표 및 목표를 포함하고 있는가?
6		상위 수준의 명세서 (예: 시스템 요구사항 명세서)가 존재하는 경우, 이와 일관되는가?
7	5.1.3 Definitions, acronyms, and abbreviations (1.3 of the SRS)	SRS를 올바르게 해석하기 위해 필요한 모든 용어, 약어 및 약어의 정의를 제공하고 있는가?
8		이 정보는 SRS의 한 개 이상의 부록에 참조로서 제공되거나 다른 문서에 참조로 제공되는가?
9	5.1.4 References (1.4 of the SRS)	SRS 내에서 다른 곳에 참조된 모든 문서의 완전한 목록을 제공하고 있는가?
10		각 문서의 제목, 보고서 번호 (적용 가능한 경우), 날짜 및 발행기관으로 식별하고 있는가?
11		참조 정보를 얻을 수 있는 출처를 지정하고 있는가?
12	5.1.5 Overview (1.5 of the SRS)	SRS의 나머지 부분에는 무엇이 포함되어 있는지 설명하고 있는가?
13		SRS의 구성방식에 대해 설명하는가?
14	5.2.1.1	각 시스템 인터페이스는 나열되어 있는가?
15	System interfaces	소프트웨어의 기능은 시스템 요구사항을 달성하는 데 활용하는가?

번호	장절	점검항목
16		시스템과 일치시키기 위한 인터페이스 설명을 제공하고 있는가?
17	5.2.1.2 User interfaces	소프트웨어 제품과 사용자 간의 각 인터페이스의 논리적 특성은 소프트웨어 요구사항을 달성하는 데 필요한 구성 특성(예: 필요한 화면 형식, 페이지 또는 창 레이아웃, 보고서 또는 메뉴의 내용 또는 프로그램 가능한 기능 키의 사용 가능성)을 포함하고 있는가?
18		시스템을 사용하는 사람과의 인터페이스를 최적화하는 모든 측면은 사용자에게 시스템이 어떻게 표시될지에 대한 지침 목록으로 구성될 수 있는가?
19		이러한 요구사항은 검증 가능한가?
20		모호한 표현은 피하고 있는가?
21	5.2.1.3 Hardware interfaces	소프트웨어 제품과 시스템의 하드웨어 구성 요소 간의 각 인터페이스의 논리적 특성을 명시하고 있는가? (형상 특성 포함) - 구성 특성(포트 개수, 명령 집합 등)이 포함 - 지원해야 하는 장치 - 지원 방법 및 프로토콜과 같은 사항
22	5.2.1.4 Software interfaces	다른 필수 소프트웨어 제품(예: 데이터 관리 시스템, 운영 체제, 수학 패키지 등)의 사용과 다른 응용 시스템과의 인터페이스(예: 채널 시스템과 일반 원장 시스템 사이의 연결)를 명시하고 있는가?
23		필수 소프트웨어 제품에 대해 다음이 제공되는가? - 이름; - 연상기호; - 사양 번호; - 버전 번호; - 출처.
24		소프트웨어 제품과 관련된 인터페이스 소프트웨어의 목적에 대한 설명을 제공하는가?
25		메시지 내용 및 형식으로 정의된 인터페이스가 제공하는가?
26		인터페이스 이름 지정 문서에 대한 참조를 제공하는가?
27	5.2.1.5	로컬 네트워크 프로토콜 등과 같은 통신에 대한 다양한 인터페

번호	장절	점검항목
	Communications interfaces	이스를 명시하고 있는가?
28	5.2.1.6 Memory constraints	주 기억장치와 보조 기억장치에 적용 가능한 특성과 한계를 명시하고 있는가?
29	5.2.1.7 Operations	사용자가 요구하는 일반 및 특수 작업을 명시하고 있는가? a) 사용자 조직에서의 다양한 작업 모드 (예: 사용자 시작 작업) b) 상호 작용 작업 및 비대면 작업의 기간 c) 데이터 처리 지원 기능 d) 백업 및 복구 작업
30	5.2.1.8 Site adaptation	특정 사이트, 임무 또는 운영 모드에 특화된 데이터 또는 초기화 순서에 대한 요구사항을 정의하고 있는가? (예: 그리드 값, 안전 한계 등)
31	requirements	특정 설치에 소프트웨어를 적용하기 위해 수정되어야 할 사이트 또는 미션 관련 기능을 지정하고 있는가?
32	5.2.2 Product	소프트웨어가 수행할 주요 기능에 대한 요약을 제공하고 있는가?
33	functions (2.2 of the SRS)	기능은 고객 또는 이 문서를 처음 읽는 사람이 이해할 수 있는 방식으로 구성되어 있는가?
34		텍스트 또는 그래픽 방법을 사용하여 서로 다양한 함수 및 관계를 표시하고 있는가?
35	5.2.3 User characteristics (2.3 of the SRS)	제품의 의도된 사용자의 일반적 특성을 설명해야 하고 있는가? (교육 수준, 경험 및 기술적 전문성 포함)
36		구체적인 요구사항을 명시하는 데 사용되어서는 안 되며, 대신 SRS에서 특정 요구사항이 나중에 명시되는 이유를 제공하고 있는가?
37	5.2.4 Constraints (2.4 of the SRS)	개발자의 옵션을 제한하는 다른 항목들에 대해 일반적인 설명을 제공하고 있는가 ? a) 규제 정책 b) 하드웨어 제한 사항 (예: 신호 타이밍 요구사항)

번호	장절	점검항목
		c) 다른 응용 프로그램과의 인터페이스 d) 병렬 운영 e) 감사 기능 f) 제어 기능 g) 고급 언어 요구사항 h) 신호 핸드셰이크 프로토콜 (예: XON-XOFF, ACK-NACK) i) 신뢰성 요구사항 j) 응용 프로그램의 중요도 k) 안전 및 보안 고려 사항
38	5.2.5 Assumptions and dependencies (2.5 of the SRS)	SRS에 명시된 요구사항에 영향을 미치는 각 요인들을 나열하고 있는가?
39	5.2.6 Apportioning of requirements (2.6 of the SRS)	시스템의 향후 버전으로 지연될 수 있는 요구사항들을 식별하고 있는가?
40	5.3 Specific requirements (Section 3 of the SRS)	시스템 요구사항을 상세하게 기술하여, 설계자가 요구사항을 충족하는 시스템 설계가 가능한가?
41		시스템 요구사항을 상세하게 기술하여, 테스터가 시스템이 요구사항을 만족하는지 테스트할 수 있는가?
42		명시된 모든 요구사항은 사용자, 운영자 또는 다른 외부 시스템에 의해 외부적으로 인지 가능한가?
43		요구사항은 적어도 시스템에 입력(자극)과 출력(응답)의 모든 설명과 입력 또는 출력 지원을 위해 시스템이 수행하는 모든 기능을 포함하고 있는가?
44		요구사항은 모든 특성을 준수하여 명시되고 있는가? a) Correct b) Unambiguous c) Complete

번호	장절	점검항목
		d) Consistent e) Ranked for importance and/or stability f) Verifiable g) Modifiable h) Traceable
45		요구사항은 관련된 이전 문서와 상호 참조되는가?
46		모든 요구사항은 고유하게 식별 가능한가?
47		요구사항을 최대한 가독성을 높이도록 조직화하는 데 주의를 기울이고 있는가?
48		소프트웨어 시스템에 대한 모든 입력과 출력에 대해 상세한 설명이 제공되는가?
49	5.3.1 External interfaces	다음과 같은 내용과 형식을 포함하고 있는가? a) 품목명 b) 목적에 대한 설명 c) 입력의 소스 또는 출력의 대상 d) 유효 범위, 정확도 및/또는 허용 범위 e) 측정 단위 f) 타이밍 g) 다른 입력/출력과 관계 h) 화면 형식/구성 i) 창 형식/조직 j) 데이터 형식 k) 명령 형식 l) 종료 메시지
50		기능 요구사항은 소프트웨어에서 입력을 받고 처리하며 출력을 처리하고 생성하는 기본적인 동작들을 정의하고 있는가?
51	5.3.2 Functions	요구사항은 일반적으로 "시스템은..."로 시작하는 "shall" 문장으로 나열되며, 다음이 포함되어 있는가? a) 입력에 대한 유효성 검사 b) 정확한 동작 순서 c) 비정상적인 상황에 대한 대응 1) 오버플로우 2) 통신 기능 3) 오류 처리 및 복구 d) 매개변수의 영향

번호	장절	점검항목
		e) 다음을 포함한 입력과 출력의 관계 1) 입력/출력 순서 2) 입력을 출력으로 변환하는 공식
52	5.3.3 Performance requirements	소프트웨어에 배치되거나 소프트웨어 전체와 사용자간 상호작용에 대한 정적 및 동적 수치 요건을 모두 명시하고 있는가? a) 지원해야 할 터미널의 수 b) 동시에 지원해야 할 사용자 수 c) 처리해야 할 정보의 양과 유형
53	5.3.4 Logical database requirements	데이터베이스에 저장될 모든 정보에 대한 논리적 요구사항을 명시하고 있는가? a) 다양한 기능에서 사용되는 정보의 유형 b) 사용 빈도 c) 접근 가능성 d) 데이터 개체와 그 관계 e) 무결성 제약 f) 데이터 보존 요구사항
54	5.3.5 Design constraint	다른 표준, 하드웨어 제한 등에 의해 부과될 수 있는 설계 제약사항을 명시하고 있는가?
55	5.3.5.1 Standards compliance	기존의 표준이나 규정에서 파생된 요구사항을 명시하고 있는가? a) 보고서 형식 b) 데이터 명명 c) 회계 절차 d) 감사 추적
56	5.3.6 Software system attributes	요구된 속성이 명시되고, 이를 객관적으로 검증할 수 있는가?
57	5.3.6.1 Reliability	납품 시 소프트웨어 시스템에 필요한 신뢰성을 확립하는 데 필요한 요소들을 명시하고 있는가?
58	5.3.6.2 Availability	체크포인트, 복구 및 재시작과 같이 전체 시스템에 정의된 가용성 수준을 보장하는 데 필요한 요소들을 명시하고 있는가?
59	5.3.6.3 Security	소프트웨어를 우발적이거나 악의적인 접근, 사용, 수정, 파괴 또는 노출로부터 보호하는 데 필요한 요소들을 명시하고 있는가?

번호	장절	점검항목
		a) 특정 암호화 기술의 활용 b) 특정 로그 또는 기록 데이터 세트 유지 c) 특정 기능을 다른 모듈에 할당 d) 프로그램의 일부 영역 사이의 통신 제한 e) 중요 변수들의 데이터 무결성 확인
60	5.3.6.4	소프트웨어의 유지보수 용이성과 관련된 소프트웨어의 특성들을 명시하고 있는가?
61	Maintainability	특정 모듈성, 인터페이스, 복잡성 등에 대한 요구사항은 배치되지 않았는가?
62	5.3.6.5	소프트웨어를 다른 호스트 기기나/또는 운영 체제로 쉽게 이식할 수 있는 소프트웨어의 특성들을 명시하고 있는가? a) 호스트 종속적인 코드를 갖는 구성 요소의 비율 b) 호스트 종속적인 코드의 비율 c) 검증된 이식 가능한 언어의 사용 d) 특정 컴파일러 또는 언어 서브셋의 사용 e) 특정 운영 체제의 사용
63	Portability	
	5.3.7	시스템의 세부요건을 최적으로 이해하기 위해 아래의 요구사항(5.3.7.1~5.3.7.7)들을 따르고 있는가?
64	Organizing the specific requirements	
64	5.3.7.1	시스템 모드를 구성할때, 인터페이스와 성능 모드에 따라 A.1 또는 A.2의 개요를 사용하여 구성되었는가?
65	System mode	
65	5.3.7.2	사용자 클래스별로 구성할때, A.3의 개요를 사용하여 구성되었는가?
66	User class	
66	5.3.7.3	객체별로 구성할때, A.4의 개요를 사용하여 구성되었는가?
67	Objects	
67	5.3.7.4	기능 특징별로 구성할때, A.5의 개요를 사용하여 구성되었는가?
68	Feature	
68	5.3.7.5	자극에 따른 절을 구성할때, A.6의 개요를 사용하여 구성되었는가?
69	Stimulus	
69	5.3.7.6	응답 생성을 지원하는 모든 기능을 구성할때, A.6의 개요를 사용하여 구성되었는가?
70	Response	
70	5.3.7.7	기능 계층별로 구성할때, A.7의 개요를 사용하여 구성되었는가?

번호	장절	점검항목
	Functional hierarchy	
71	5.3.8 Additional comments	사양에 따라 시스템의 특정 요구에 맞게 조정된 여러 계층 구조에 대한 사양 cc 요구사항을 구성하고 있는가?
72	5.4 Supporting information	지원 정보는 다음의 내용을 포함하고 있는가? a) 목차 b) 색인 c) 부록
73	5.4.1 Table of contents and index	목차와 색인은 매우 중요하며 일반적인 구성 관행을 따르고 있는가?
74	5.4.2 Appendixes	부록에는 다음과 같은 내용이 포함되는가? a) 샘플 입력/출력 형식, 비용 분석 연구 설명 또는 사용자 설문 결과 b) SRS의 독자들에게 도움이 될 수 있는 지원 또는 배경 정보 c) 소프트웨어로 해결해야 할 문제에 대한 설명 d) 보안, 내보내기, 초기 로딩 또는 기타 요구사항을 충족하기 위한 코드와 미디어의 특수 포장 지시사항
75		부록이 포함된 경우, SRS는 명시적으로 부록이 요구사항의 일부로 간주되는지 여부를 명시하고 있는가?

표 11. 소프트웨어 V&V 체크리스트 - IEEE 830

소프트웨어 V&V 체크리스트 - IEEE 1016

번호	장절	점검항목
1	1.1 Scope	표준은 소프트웨어 설계를 설명하고 소프트웨어 설계 설명서 (SDD)의 정보 내용과 구성을 정의하고 있는가?
2		SDD는 설계 정보를 기록하고 해당 정보를 주요 설계 이해관계자에게 전달하는 데 사용되는가?
3		표준은 SDD에서 설계 언어를 선택하는 데 요구사항을 정의하고 있는가?
4	1.2 Purpose	표준은 SDD의 정보 내용과 구성에 대한 요구사항을 명시하는가?
5		표준은 SDD에 사용할 설계 언어의 선택에 대한 요구사항과 SDD를 구성하는 데 사용할 설계 뷰를 문서화하는 요구사항을 명시하고 있는가?
6	1.3 Intended audience	표준은 SDD를 준비하고 사용하는 기술 및 관리 이해관계자들의 대상인가?
7		표준은 설계자에게 설계 정보의 선택, 구성 및 제시에 대한 가이드를 제공하는가?
8		설계 설명이 완전하고 간결하며 일관되고 상호 교환 가능하며, 학습된 설계 경험과 교훈을 기록하는 데 적합하고, 잘 구성되어 있으며, 의사소통이 용이하도록 보장되는가?
9	1.4 Conformance	SDD가 이 표준의 4절과 5절에 있는 모든 요구사항을 만족하고 있는가?
10	3. Conceptual model for software design descriptions	개념적 모델은 SDD의 기본 용어와 개념, SDD가 준비되고 사용되는 문맥, SDD를 사용하는 이해관계자들, 또한 그들이 SDD를 사용하는 방법이 포함되는가?
11	3.1 Software design in context	설계는 설계 대상(설계 중인 시스템 또는 소프트웨어)의 개념화로, 필수적인 특성을 담고 있는가?
12		설계는 요구사항을 충족시키는 방법을 보여주고 있는가?
13		설계는 분석과 평가의 기초로 하는가?
14		설계는 구현을 지도하는 데 사용되는가?
15		SDD는 설계 뷰를 사용하여 구성되고 있는가?

번호	장절	점검항목
16		각 설계 뷰는 설계 관점에 의해 제어되고 있는가?
17		설계에는 대안에 대한 고려와 평가, 의사결정으로 이어지는 대안 사이의 절충이 포함되어 있는가?
18	3.2 Software design descriptions within the life cycle	이 표준의 생명주기는 IEEE Std 12207-2008 [B21]을 기반으로 하고 있는가?
19	3.2.1 Influences on SDD preparation	SRS는 설계를 주도할 소프트웨어 요구사항을 포착하며 고려해야 할 설계 제약사항을 기술하고 있는가? (IEEE Std 830™-1998 [B15] 및 ISO/IEC 15289:2006 [B25] 참조).
20	3.2.2 influences on software life cycle products	요구사항과 설계 사이의 추적성은 유지되고 있는가? (Abran과 Moore [B1] 참조).
21		테스트 사례 및 테스트 절차 개발 시 SDD 내용을 고려하고 있는가?
22	3.2.3 Design verification and design role in validation	설계는 이해관계자들의 설계 관심사를 해결하고 있는가?
23		설계는 명시된 요구사항과 일치하는가?
24		설계는 인터페이스, 입력, 출력, 알고리즘, 자원 할당 및 오류 처리와 관련된 의도된 설계 결정을 구현하는가?
25		설계는 안전성, 보안성 또는 유지보수성과 같은 의도된 품질을 달성하는가?
26		설계는 정해진 아키텍처를 준수하고 있는가?
27		SDD는 검증에서 역할을 수행하며, 다음과 같은 정보를 제공하고 있는가? a. 구현을 이해하기 위해 필요한 개요 b. 설계 결정을 정당화하는 이유 c. 소프트웨어 설계의 요구사항들과의 추적성 (검증과 검증에 대한 설명은 IEEE Std 1012™-2004 [B16] 참조.)
28	4.1 Introduction	SDD는 다음의 필수 내용을 정의하고 있는가? - SDD 식별

번호	장절	점검항목
		<ul style="list-style-type: none"> - 식별된 설계 이해관계자 - 식별된 설계 관심사 - 선택된 설계 관점, 각각의 허용된 설계 요소와 설계 언어에 대한 유형 정의 - 설계 뷰 - 설계 오버레이 - 설계 근거
29	4.2 SDD identification	<p>SDD는 다음과 같은 기술적 정보를 포함하고 있는가?</p> <ul style="list-style-type: none"> - 발행일 및 현황 - 범위 - 발행 기관 - 저작자 (책임 또는 저작권 정보) - 참조 문헌 - 문맥 - 각 설계 뷰포인트에 사용되는 하나 이상의 설계 언어 - 본문 - 요약 - 용어집 - 변경 이력
30	4.3 Design stakeholders and their concerns	SDD는 설계 주제에 대한 설계 이해관계자를 식별하고 있는가?
31		SDD는 각 식별된 설계 이해관계자의 설계 문제를 식별하고 있는가?
32		SDD는 식별된 각 설계 문제를 해결하였는가?
33	4.4 Design views	각 설계 뷰는 해당 설계뷰 뷰포인트에 의해 명시된 설계 문제를 해결하고 있는가?
34		하나 이상의 설계 뷰로 구성된 SDD가 존재하는가?
35		SDD 내 각 설계 뷰는 해당되는 설계 관점을 따르고 있는가?
36		각 설계 뷰는 해당되는 설계 관점에 의해 명시된 설계 관련 문제를 다루고 있는가?
37		모든 식별된 설계 관련 문제가 적어도 하나의 설계 뷰의 주제가 되도록 하는가?
38		각 관점에서 세분화된 모든 설계 속성은 해당 뷰와 관련된 모든 설계 개체와 관계에 대해 지정되어 있는가?

번호	장절	점검항목
39		모든 설계 제약 조건이 적용되어 있는가?
40	4.5 Design viewpoints	<p>SDD의 각 설계 뷰에는 정확하게 하나의 설계 관점이 지배되어야 하며, 각 설계 관점은 다음과 같이 명시되어 있는가?</p> <ul style="list-style-type: none"> - 뷰 포인트의 이름 - 뷰 포인트의 주제가 되는 설계 관련 문제 - 뷰 포인트에 의해 정의된 설계 요소들, 구체적으로 설계 개체, 속성, 관계, 그리고 해당 관점에 의해 도입되거나 해당 관점에서 사용되는 제약 사항들 (이는 다른 곳에서 정의되었을 수도 있음). - 뷰 포인트에 기반하여 설계 뷰를 구성하는 데 사용되는 분석 방법 또는 기타 작업들, 그리고 설계를 해석하고 평가하는 기준 - 해당 뷰 포인트의 출처 (예: 저작자 또는 인용)
41		<p>설계 뷰 포인트 사양은 뷰 포인트 사용에 대한 다음의 정보를 제공하고 있는가?</p> <ul style="list-style-type: none"> -뷰 포인트에 적용되는 형식적 또는 비형식적 일관성과 완결성 테스트 -뷰 포인트에 적용되는 평가 또는 분석 기법 -뷰 포인트의 구성 또는 합성을 돕기 위한 휴리스틱, 패턴 또는 지침들
42		SDD에는 선택된 각 시점을 선택하기 위한 근거를 포함하고 있는가?
43		SDD에서 식별된 각 설계문제는 사용을 위해 선택된 하나 이상의 설계 뷰포인트로 구성되어 있는가?
44	4.6 Design elements	각 설계 요소의 유형은 정확히 하나의 뷰 포인트 정의 내에서 도입되어 있는가?
45		SDD의 각 설계 구성요소에는 이름, 형식 및 내용을 가지고 있는가?
46		설계 요소는 설계 엔티티, 설계 관계, 설계 속성 및 설계 제약 조건의 네 가지 하위 사례에 의해 "상속"되어 있는가?
47		설계 요소는 관련 뷰포인트 내에서 형식 정의에 따라 정확히 하나의 설계 뷰포인트에 의해 도입되고 "소유"되고 있는가?
48	4.6.2.1 Name attribute	각 설계 요소의 이름은 명확한 기준의 이름을 가지고 있는가?
49	4.6.2.2 Type	형식 속성은 요소의 특성을 설명하고 있는가?

번호	장절	점검항목
	attribute	<ul style="list-style-type: none"> - 서브 시스템 - 컴포넌트 - 프레임워크 - 라이브러리 - 클래스 - 서브 프로그램 모듈 - 프로그램 유닛 - 함수 - 절차 - 프로세스 - 객체 - 영구 객체 - 데이터 저장소 등
50		SDD 내에서 선택된 요소 특성은 일관되게 적용되어 있는가?
51	4.6.2.3 Purpose attribute	목적 속성은 해당 요소가 생성된 근거를 제공하고 있는가?
52	4.6.2.4 Author attribute	작성자 속성은 해당 요소의 설계자를 식별하고 있는가?
53	4.6.3 Design relationships	설계 관계는 두 개 이상의 설계 개체 사이의 연결 또는 대응을 지칭하고, 문서는 해당 설계 개체들에 대한 사실을 제공하고 있는가?
54		SDD 내의 각 설계 관계는 이름(4.6.2.1)과 유형(4.6.2.2)을 가지고 있는가?
55		설계 관계는 관련된 설계 개체들을 식별하고 있는가?
56	4.6.4 Design constraints	SDD 내의 각 설계 제약 조건은 이름(4.6.2.1)과 유형(4.6.2.2)을 가지고 있는가?
57		설계 제약 조건은 출처와 대상 설계 개체를 식별하고 있는가?
58	4.7 Design overlays	각 설계 오버레이는 고유하게 이름을 지정하고 오버레이로 표시되는가?
59		각 설계 오버레이는 단일 뷰 포인트와 명확하게 연관되어 있는가?
60	4.8 Design	설계 근거는 시스템 설계를 이끈 설계자의 추론과 해당 결정들을

번호	장절	점검항목
	rationale	<p>정당화하는 내용을 담고 있는가?</p> <ul style="list-style-type: none"> - 설계 관련 문제들과 설계 요구사항에 대한 대응으로 제기되고 해결된 사항 - 고려된 설계 옵션 - 절충점 - 결정 사항 - 설계 결정을 이끌기 위해 사용된 기준 - 결정에 이르기 위해 제시된 주장과 정당화
61	4.9 Design languages	설계 언어는 설계 뷰 포인트 사양(4.5)의 일부로 선택되는가?
62		<p>설계 언어는 다음과 같은 요소들을 제공하는가?</p> <ul style="list-style-type: none"> - 명확히 정의된 구문과 의미론 - 사용 가능한 표준 또는 동등한 정의 문서의 상태 - SDD에서는 표준화되고 잘 확립된(즉, 이전에 정의되고 편리하게 사용 가능한) 설계 언어
63		새롭게 발명된 설계 언어의 경우, 언어 정의는 뷰 포인트 선언의 일부로 제공되고 있는가?
64	5.1 Introduction	각 뷰 포인트에 대해 이름, 설계 관련 문제들과 적절한 설계 언어들이 목록으로 제공되고 있는가?
65		각 뷰 포인트에는 최소한의 설계 개체, 설계 관계, 설계 개체 속성 및 설계 제약 조건과 관련된 간단한 설명이 제공되고 있는가?
66		정의된 설계 뷰 포인트는 식별된 설계 관련 문제들(4.3)을 기반으로 설계 대상에 적용 가능한 경우에만 SDD에서 사용되고 있는가?
67	5.2 Context viewpoint	Context 뷰 포인트의 문맥은 사용자와 다른 이해 관계자들을 포함하여 설계 대상이 환경에서 상호 작용하는 주체들을 기준으로 정의되어 있는가?
68		Context 뷰 포인트는 설계 대상에 대한 "블랙 박스" 관점을 제공하고 있는가?
69		서비스의 계층화와 행위자와 시스템과의 상호 작용에 대한 시나리오 형태로 된 설명은 세부 정보를 추가하는 메커니즘을 제공하고 있는가?
70		컨텍스트 보기에 대한 배포 오버레이는 실행 하드웨어 플랫폼이 설계 제목의 일부일 때마다 배포 보기로 변환될 수 있는가?
71		독립 실행형 소프트웨어 설계의 경우, 배포 오버레이는 소프트웨어

번호	장절	점검항목
		어 개체를 현재의 설계 작업의 대상이 아닌 외부에서 사용 가능한 개체에 매핑하고 있는가?
72	5.3 Composition viewpoint	구성 뷰 포인트는 설계 대상이 (재귀적으로) 구성 요소들로 구조화되는 방식을 설명하고, 해당 요소들의 역할을 정의하고 있는가?
73	5.4 Logical viewpoint	논리적 뷰 포인트는 유형의 인스턴스 예제를 사용하여 설계 아이디어를 개요화하고 있는가?
74	5.5 Dependency viewpoint	종속성 뷰 포인트는 개체들 간의 상호 연결과 접근 관계를 지정하고 있는가? - 공유 정보 - 실행 순서 또는 인터페이스의 매개변수화 등이 포함
75	5.6 Information viewpoint	정보 뷰 포인트는 설계 대상과 관련하여 상당한 영구 데이터 내용이 예상될 때 적용되는가?
76	5.7 Patterns use viewpoint	이 뷰 포인트는 추상화된 역할과 커넥터를 포함하는 협업 패턴으로서의 설계 아이디어(생겨난 개념)을 다루고 있는가?
77	5.8 Interface viewpoint	인터페이스 뷰 포인트는 설계 대상이 제공하는 서비스를 올바르게 사용하는 방법을 정보 설계자, 프로그래머, 테스터들에게 제공하고 있는가?
78		인터페이스 뷰 포인트의 설명은 SRS에서 제공되지 않는 외부 및 내부 인터페이스의 세부 정보들이 포함되어 있는가?
79		인터페이스 뷰 포인트는 각 개체에 대한 인터페이스 사양의 집합으로 구성되어 있는가?
80	5.9 Structure viewpoint	구조 뷰 포인트는 설계 대상의 내부 구성 요소와 구성을 유사한 요소들로 (재귀적으로) 문서화하는 데 사용되고 있는가?
81	5.10 Interaction viewpoint	상호작용 뷰 포인트는 개체들 간의 상호작용에 대한 전략을 정의하고 있는가? - 어떤 이유로 - 어디에서 - 어떻게 - 그리고 - 어떤 수준에서 행동이 발생하는지
82	5.11 State	반응형 시스템과 내부 동작이 관심사인 시스템은 이 관점을 사용

번호	장절	점검항목
	dynamics viewpoint	하고 있는가?
83	5.12 Algorithm viewpoint	각각의 설계 개체의 작업(예: 메서드와 함수)에 대한 상세한 설계 설명과 ,내부 세부 사항 및 논리를 포함하고 있는가?
84	5.13 Resource viewpoint	자원 뷰 포인트의 목적에 따라, 설계 대상의 특성과 자원 활용을 모델링하고 있는가?
85	5.13.1 Design concerns	주요 관심사는 리소스 활용, 리소스 경합, 가용성 및 성능으로 맞춰지는가?

표 12. 소프트웨어 V&V 체크리스트 - IEEE 1016

부록 B. 참조 문서

본 문서에 적용되는 참고문서는 다음과 같다. 참고문서에서 필요한 부분만 인용되었으며, 별도의 기술이 없다면 최근 개정본이 적용된다.

- 1) KEPIC EME 3100 [IEEE Std. 1012-2004], “IEEE Standard for Software Verification and Validation”
- 2) KEPIC EME 3200 [IEEE Std. 830-1998], “IEEE Recommended Practice for Software Requirements Specifications”
- 3) [IEEE Std. 1016], “IEEE Standard for Information Technology -Systems Design- Software Design Descriptions”
- 4) [IEEE Std. 1471], “IEEE Recommended Practice for Architectural Description of Software-Intensive Systems”
- 5) [KC 62619], “전기용품안전기준”
- 6) [IEC 61508], “Functional safety of electronic safety-related systems”

부록 C. 코딩 규칙

MISRA-C:2012

규칙	제목
MISRA-C:2012_01_01	프로그램은 C 표준과 사용하는 컴파일러의 번역 제한을 준수해야 함
MISRA-C:2012_01_02	언어 확장을 사용하면 안 됨
MISRA-C:2012_01_03	정의되지 않거나 명시되지 않은 행동이 발생하면 안 됨
MISRA-C:2012_02_01	도달할 수 없는 코드(unreachable code)가 있으면 안 됨
MISRA-C:2012_02_02	죽은 코드(dead code)는 없어야 함
MISRA-C:2012_02_03	사용되지 않은 타입 선언은 없어야 함
MISRA-C:2012_02_04	사용되지 않은 tag(struct, union, enum) 선언은 없어야 함
MISRA-C:2012_02_05	사용되지 않은 매크로 선언은 없어야 함
MISRA-C:2012_02_06	함수 내에 사용되지 않은 레이블(label) 선언은 없어야 함
MISRA-C:2012_02_07	함수에서 사용되지 않은 파라미터는 없어야 함
MISRA-C:2012_03_01	문자열 /*와 //는 주석 안에서 사용하지 말아야 함
MISRA-C:2012_03_02	행 접합(Line-splicing)은 // 주석 내에서 사용하면 안 됨
MISRA-C:2012_04_01	8진수, 16진수 escape 시퀀스에 다른 escape 시퀀스 외에는 붙이면 안 됨
MISRA-C:2012_04_02	Trigraph는 사용하면 안 됨
MISRA-C:2012_05_01	외부 식별자는 구별되어야 함
MISRA-C:2012_05_02	같은 scope 혹은 name space에 선언된 식별자는 구별되어야 함
MISRA-C:2012_05_03	안쪽 scope의 식별자를 바깥 scope의 식별자가 가리면 안 됨
MISRA-C:2012_05_04	매크로의 식별자는 구별되어야 함
MISRA-C:2012_05_05	식별자는 매크로 이름과 구별되어야 함
MISRA-C:2012_05_06	typedef 이름은 유일한 식별자여야 함
MISRA-C:2012_05_07	tag(struct, union, enum) 이름은 유일한 식별자여야 함
MISRA-C:2012_05_08	외부 연결을 갖는 변수나 함수 식별자는 유일해야 함
MISRA-C:2012_05_09	내부 연결을 갖는 변수나 함수 식별자는 유일해야 함

규칙	제목
MISRA-C:2012_06_01	bit-field는 올바른 타입으로 선언되어야 함
MISRA-C:2012_06_02	single-bit로 표현된 이름있는 bit-field는 signed면 안 됨
MISRA-C:2012_07_01	8진수 상수는 사용하면 안 됨
MISRA-C:2012_07_02	unsigned integer 상수에는 접미사 "u"나 "U"를 붙여야 함
MISRA-C:2012_07_03	소문자 접미사 "l" 은 사용하면 안 됨
MISRA-C:2012_07_04	문자열을 const char * 타입이 아닌 객체에 할당하면 안 됨
MISRA-C:2012_08_01	타입은 명시적으로 입력해야 함
MISRA-C:2012_08_02	함수는 이름 있는 파라미터로 구성된 프로토타입 형태여야 함
MISRA-C:2012_08_03	동일한 객체 또는 함수의 모든 선언은 같은 이름과 타입 한정자를 사용해야 함
MISRA-C:2012_08_04	객체나 함수의 정의 또는 호출 이전에 호환가능한 선언이 존재해야 함
MISRA-C:2012_08_05	외부연결(external linkage)를 갖는 객체나 함수는 오직 하나의 파일에서만 선언되어야 함
MISRA-C:2012_08_06	외부연결 식별자는 하나의 외부정의(external definition)을 가져야 함
MISRA-C:2012_08_07	함수나 객체가 하나의 번역단위에서 참조된다면 외부참조(external linkage)로 정의되면 안됨
MISRA-C:2012_08_08	내부 연결(internal linkage) 을 갖는 모든 객체 또는 함수는 static 을 이용해야 함
MISRA-C:2012_08_09	단일 함수에만 쓰이는 객체는 해당 블록범위에서 정의되어야 한다.
MISRA-C:2012_08_10	인라인 함수는 static으로 선언되어야 한다.
MISRA-C:2012_08_11	배열에 대한 외부연결(external linkage) 선언 시, 그 크기를 명시적으로 해야 함
MISRA-C:2012_08_12	열거자 리스트 내에서 묵시적으로 지정된 열거형 상수의 값은 유일해야 함
MISRA-C:2012_08_13	가능하다면 포인터는 const 로 한정된 타입을 가리켜야 함
MISRA-C:2012_08_14	restrict 사용금지
MISRA-C:2012_09_01	모든 변수는 읽기 전에 할당되어야 함

규칙	제목
MISRA-C:2012_09_02	배열, 구조체, 유니온 타입의 초기화는 큰괄호('{ }') 로 둘러쌓여야 함
MISRA-C:2012_09_03	배열은 일부분만 초기화 되면 안됨
MISRA-C:2012_09_04	객체의 요소 두 번 이상 초기화 금지
MISRA-C:2012_09_05	지정 초기화(designated initializer)가 사용된 배열은 그 크기가 명시적이어야 함
MISRA-C:2012_10_01	부적절한 essential 타입의 피연산자를 사용하지 않아야 함
MISRA-C:2012_10_02	Essential 타입이 character인 표현식은 가감연산자에 부적합하게 사용되지 않아야 함
MISRA-C:2012_10_03	표현식의 값은 더 작은 essential 타입이나 다른 essential 타입 분류에 타입을 갖는 객체에 할당되지 않아야 함
MISRA-C:2012_10_04	일반 산술 변환이 수행되는 연산자의 두 피연산자들은 필히 같은 essential 타입 분류에 속하는 타입이어야 함
MISRA-C:2012_10_05	수식의 값은 적절하지 않은 essential type으로 변환되지 않아야 함
MISRA-C:2012_10_06	복합 표현식의 값은 더 큰 essential 타입의 객체에 할당되지 않아야 함
MISRA-C:2012_10_07	복합 수식이 기본 산술 변환을 수행하는 연산자의 피연산자로 사용된 경우, 다른 피연산자는 해당 수식의 타입보다 큰 essential 타입을 가지지 않아야 함
MISRA-C:2012_10_08	복합 수식의 값은 다른 essential 타입 분류에 속하는 타입이나 더 큰 essential 타입으로 변환되지 않아야 함
MISRA-C:2012_11_01	함수 포인터와 다른 타입 간의 형 변환 금지
MISRA-C:2012_11_02	불완전 포인터(incomplete)타입과 다른 타입 간의 형 변환 금지
MISRA-C:2012_11_03	한 객체 포인터로와 다른 객체 포인터 간의 변환 금지
MISRA-C:2012_11_04	객체 포인터와 정수형 타입 간의 변환 금지
MISRA-C:2012_11_05	void 포인터에서 객체 포인터로 변환 금지
MISRA-C:2012_11_06	void 포인터와 산술 타입 간의 변환 금지
MISRA-C:2012_11_07	객체 포인터와 정수가 아닌 산술 타입 간의 변환금지
MISRA-C:2012_11_08	형 변환 시 포인터의 const 또는 volatile 제거금지

규칙	제목
MISRA-C:2012_11_09	매크로 NULL은 정수 null 포인터 상수만으로 사용해야 함
MISRA-C:2012_12_01	수식 내부에 사용된 연산자의 우선순위가 명확한지 검사
MISRA-C:2012_12_02	shift 연산자의 우측 피연산자가 좌측 피연산자의 essential 타입의 범위 내의 정수여야 함
MISRA-C:2012_12_03	Comma 연산자 사용금지
MISRA-C:2012_12_04	unsigned integer wrap-around를 발생시키는 상수 수식 금지
MISRA-C:2012_13_01	초기화 리스트가 영구적인 side effect를 일으키면 안 됨
MISRA-C:2012_13_02	수식의 값과 영구적인 side effect의 평가결과가 평가 순서에 상관없이 같아야 함
MISRA-C:2012_13_03	증감 연산자(++ , --)를 포함한 수식은 해당 증감 연산자를 제외한 잠재적인 side effect 금지
MISRA-C:2012_13_04	할당 연산자의 결과를 사용 금지
MISRA-C:2012_13_05	논리적 &&나 연산자의 우측 항에 영구적인 side effect 가 있으면 안 됨
MISRA-C:2012_13_06	sizeof의 피연산자에 side effect를 발생시킬 수 있는 수식 포함 금지
MISRA-C:2012_14_01	loop counter 의 essential 타입이 실수형이면 안 됨
MISRA-C:2012_14_02	for 문이 잘 짜여져야 함
MISRA-C:2012_14_03	결과가 항상 같은 제어식 사용 금지
MISRA-C:2012_14_04	조건문이나 반복문의 제어식이 essentially Boolean type 인지 검사
MISRA-C:2012_15_01	goto 문 사용 금지
MISRA-C:2012_15_02	goto 문의 label이 같은 함수에서 더 나중에 위치하는지 검사
MISRA-C:2012_15_03	goto 문이 참조하는 label은 같은 블록이나 인접한 블록에 있는지 검사
MISRA-C:2012_15_04	반복문에는 하나의 break 나 goto 문만 있는지 검사
MISRA-C:2012_15_05	함수는 마지막에 하나의 return만 가지는지 검사
MISRA-C:2012_15_06	반복문이나 선택문이 복합문인지 검사
MISRA-C:2012_15_07	모든 if ... else if 구조는 else 문으로 끝나는지 검사

규칙	제목
MISRA-C:2012_16_01	모든 switch 문은 잘 짜여야 함
MISRA-C:2012_16_02	switch label 을 포함하는 가장 가까운 문장은 switch 문이어야 함
MISRA-C:2012_16_03	모든 switch 절은 break 로 끝나야 함
MISRA-C:2012_16_04	모든 switch 문에 default label 이 있어야 함
MISRA-C:2012_16_05	default label 은 switch 문장의 맨처음이나 마지막 switch label 이어야 함
MISRA-C:2012_16_06	모든 switch 문에 적어도 둘 이상의 switch 절이 있어야 함
MISRA-C:2012_16_07	switch-expression 은 Boolean 타입이면 안 됨
MISRA-C:2012_17_01	stdarg.h에 정의된 요소들은 사용 금지
MISRA-C:2012_17_02	직, 간접적 재귀호출 금지
MISRA-C:2012_17_03	함수를 묵시적으로 선언 금지
MISRA-C:2012_17_04	리턴 타입이 void가 아닌 함수의 모든 경로의 마지막은 수식을 포함한 return 문이어야 함
MISRA-C:2012_17_05	배열 타입으로 선언된 함수의 매개변수는 적합한 수의 원소를 가져야 함
MISRA-C:2012_17_06	배열 타입의 매개 변수의 선언은 [] 사이에 static keyword를 포함하지 않아야 함
MISRA-C:2012_17_07	리턴 타입이 void가 아닌 함수의 리턴 값은 필히 사용해야 함
MISRA-C:2012_17_08	함수의 매개변수는 변경되지 않아야 함
MISRA-C:2012_18_01	포인터의 연산 결과는 해당 포인터가 가리키는 배열의 요소이어야 함
MISRA-C:2012_18_02	포인터 간의 뺄셈은 같은 배열의 요소들을 가리키고 있는 포인터들에만 적용되어야 함
MISRA-C:2012_18_03	관계 연산자 >, >=, <, 그리고 <=는 같은 객체를 가리키고 있는 포인터를 제외한 다른 포인터 타입의 객체에 적용 금지
MISRA-C:2012_18_04	+, -, +=, -= 연산자는 포인터 타입의 표현식에 적용 금지
MISRA-C:2012_18_05	3차원 이상의 포인터 선언 금지
MISRA-C:2012_18_06	자동으로 값이 할당되는 객체의 주소는, 처음 객체가 소멸된 이후에도 지속되는 다른 객체로 복사 금지

규칙	제목
MISRA-C:2012_18_07	구조체의 멤버로써 가변 배열 선언금지
MISRA-C:2012_18_08	Variable-length(길이를 변수로 지정한) 배열 타입 사용 금지
MISRA-C:2012_19_01	오버랩되는 객체에 할당 또는 복사 금지
MISRA-C:2012_19_02	union 키워드 사용 금지
MISRA-C:2012_20_01	소스코드에서 #include의 상단에는 전처리 지시자 또는 주석만 허용
MISRA-C:2012_20_02	문자 ', ", \ 와 문자열 /* , //은 헤더파일의 이름에 포함 금지
MISRA-C:2012_20_03	#include 지시자는 또는 "filename"의 형태를 따라야 함
MISRA-C:2012_20_04	키워드와 같은 이름으로 매크로 정의 금지
MISRA-C:2012_20_05	#undef 사용 금지
MISRA-C:2012_20_06	매크로 인자 부분에 전처리 지시자 사용금지
MISRA-C:2012_20_07	매크로 인자는 괄호로 감싸야 함
MISRA-C:2012_20_08	#if 또는 #elif의 제어 표현식은 0 또는 1로 값이 나와야 함
MISRA-C:2012_20_09	#if 또는 #elif의 제어 표현식에 사용된 모든 식별자는 평가하기 전에 #define으로 정의 되어야 함
MISRA-C:2012_20_10	전처리 연산자 # 과 ## 사용금지
MISRA-C:2012_20_11	전처리 연산자 #의 피연산자 바로 뒤에 ## 연산자 사용금지
MISRA-C:2012_20_12	# 또는 ## 의 피연산자이며 추가적인 매크로 치환이 필요한 매크로 인자는 다른 연산자의 피연산자로 사용 금지
MISRA-C:2012_20_13	첫번째 토큰이 # 인 행은 유효한 전처리 지시자여야 함
MISRA-C:2012_20_14	모든 #else, #elif, #endif 전처리 지시자는 같은 파일 내에 관련된 #if, #ifdef, #ifndef 가 존재해야 한다.
MISRA-C:2012_21_01	예약된 식별자 또는 예약된 매크로 이름을 #define 또는 #undef에 사용 금지
MISRA-C:2012_21_02	예약된 지시자 또는 예약된 매크로 이름으로 선언 금지
MISRA-C:2012_21_03	stdlib.h의 메모리 할당과 해제 함수 사용 금지
MISRA-C:2012_21_04	setjmp.h 사용 금지
MISRA-C:2012_21_05	signal.h 사용 금지
MISRA-C:2012_21_06	표준 입출력 함수 사용 금지

규칙	제목
MISRA-C:2012_21_07	stdlib.h의 atof, atoi, atol, atoll 함수 사용 금지
MISRA-C:2012_21_08	stdlib.h의 라이브러리 함수인 abort, exit, getenv, system 사용 금지
MISRA-C:2012_21_09	stdlib.h의 라이브러리 함수 bsearch, qsort 사용 금지
MISRA-C:2012_21_10	표준 라이브러리 time, date 함수 사용 금지
MISRA-C:2012_21_11	tgmath.h 사용 금지
MISRA-C:2012_21_12	fenv.h의 예외 처리 식별자 사용 금지
MISRA-C:2012_22_01	표준 라이브러리를 통해 동적으로 얻은 리소스는 명시적으로 해제되어야 함
MISRA-C:2012_22_02	메모리는 시스템 라이브러리 함수를 통해 할당되었을 때만 해제되어야 함
MISRA-C:2012_22_03	서로 다른 스트림에서 같은 파일을 동시에 읽기, 쓰기로 열면 안 됨
MISRA-C:2012_22_04	읽기 전용으로 열린 스트림에 쓰기를 하면 안 됨
MISRA-C:2012_22_05	FILE 객체를 가리키는 포인터는 dereference되면 안 됨
MISRA-C:2012_22_06	FILE 객체를 가리키는 포인터를 해당 스트림이 닫힌 후에 사용하면 안 됨

표 13. 코딩 규칙 - MISRA-C:2012

ISO 26262

규칙	제목
ISO 26262_1a	함수가 하나의 exit point를 가졌는지 검사
ISO 26262_1b	동적 메모리 할당 금지 및 동적 할당된 변수에 대한 사용 전 검사
ISO 26262_1c	변수 초기화
ISO 26262_1d	변수의 이름이 중복되어 사용되면 안됨
ISO 26262_1e	전역변수 금지
ISO 26262_1f	포인터 사용 제약
ISO 26262_1g	목시적인 타입 변환 금지
ISO 26262_1h	숨겨진 데이터 흐름이나 제어 흐름 검사
ISO 26262_1i	무조건 분기 사용 금지
ISO 26262_1j	함수의 직간접적인 재귀호출 금지

표 14. 코딩 규칙 - ISO 26262

NUREG/CR-6463

규칙	제목
NUREG_001	동적 메모리 할당 금지
NUREG_002	free된 포인터 사용 금지
NUREG_003	free된 포인터에 즉시 새로운 값을 저장하였는지 검사
NUREG_004	메모리 할당 함수의 할당 성공 여부 반환값이 검증되었는지 검사
NUREG_005	클래스 연산자를 명시적으로 선언하였는지 검사
NUREG_006	메모리 할당 함수의 할당 크기를 입력하는 인자에 sizeof를 포함한 expression을 사용했는지 검사
NUREG_007	특정 함수의 특정 인자로 검증되지 않은 변수 또는 0(constant zero) 사용 금지
NUREG_008	realloc()의 인자의 타입과 cast되는 타입이 다른지 검사
NUREG_009	사용이 권장되지 않거나 구형의 함수 사용 금지
NUREG_010	malloc 함수 사용시 오버플로가 발생하면 안된다.
NUREG_011	memcpy 함수 사용시 배열에 복사 할 때 오버플로가 발생하는지 검사
NUREG_012	string 관련 함수 사용시 배열에 복사 할 때 오버플로가 발생하는지 검사
NUREG_013	함수의 파라미터 개수 제한
NUREG_014	함수의 직, 간접적인 재귀 호출 금지
NUREG_015	사용이 권장되지 않거나 구형의 함수 사용 금지
NUREG_016	Library 함수에 전달되는 인자 값이 올바른지 검사해야 함
NUREG_017	배열의 index에 범위를 벗어나는 상수를 사용했는지 검사
NUREG_018	Goto statement 사용 금지
NUREG_019	Longjmp 함수와 setjmp 매크로 사용 금지
NUREG_020	함수 복잡도(cyclomatic complexity number) 제한
NUREG_021	else if의 개수 제한
NUREG_022	If 문(else if 또는 else 포함)의 body는 복합문이어야 함
NUREG_023	switch 문의 마지막 절이 default 절이어야 함
NUREG_024	문장이 있는 모든 case 또는 default label은 조건문이 없는 break문으로 끝나야 함
NUREG_025	Switch 문의 마지막 절이 default 절이어야 함
NUREG_026	도달 불가능한 코드 사용 금지

규칙	제목
NUREG_027	변수 선언 시 초기화 여부 검사
NUREG_028	External linkage를 포함하는 identifier는 정확히 하나의 external definition을 가져야 함
NUREG_029	Automatic 공간을 사용하는 객체를 대상으로, 주소에 할당된 객체가 삭제된 이후에도 존재할 수 있는 객체에는 할당 금지
NUREG_030	reference 타입 함수에서 parameter 반환 금지
NUREG_031	변수 선언 시 초기화 여부 검사
NUREG_032	조건식으로 검사되지 않은 포인터 사용 금지
NUREG_033	null 값을 가지는 객체에 대해 dereference 금지
NUREG_034	하나의 선언문에 2개 이상의 변수선언 금지
NUREG_035	하나의 선언문에 포인터와 포인터가 아닌 객체의 선언이 혼합되었는지 검사
NUREG_036	하나의 함수는 함수의 끝에 하나의 exit point를 가져야 함
NUREG_037	Non-void return type 함수의 모든 출구는 하나의 명시적인 return 구문을 가져야 함
NUREG_038	함수의 선언과 정의에서 return type이 동일하며 변수의 type 또한 동일해야 함
NUREG_039	함수의 prototype 선언 시 모든 parameter를 위한 identifier가 주어져야 함
NUREG_040	함수에 parameter가 없는 경우 void type parameter로 선언해야 함
NUREG_041	함수 identifier는 주소 값(&가 붙은 형태)으로 사용하거나, 괄호화 된 parameter 리스트(identifier 뒤에 ()을 붙임)로 사용해야 함
NUREG_042	void 타입 함수에서 값을 가진 return문이 존재하면 안됨
NUREG_043	함수의 인자 수는 가변적으로 정의되서는 안 됨
NUREG_044	함수 호출 시 인자의 개수는 parameter의 수와 일치해야 함
NUREG_045	함수 매크로는 모든 argument를 호출해야 함
NUREG_046	포인터 타입 파라미터를 사용하기 전에 검증이 존재하는지 검사
NUREG_047	함수에서 반환되는 값이 검증되었는지 검사
NUREG_048	Expression의 값들은 표준이 허용하는 평가 순서가 같아야 함
NUREG_049	비트 필드 사용 금지

규칙	제목
NUREG_050	Basic numerical type 대신 크기와 부호를 나타내는 typedef를 사용해야 함
NUREG_051	Function pointer type과는 integer type을 제외한 어떤 type 간에도 변환이 일어나선 안 됨
NUREG_052	Object의 포인터와는 integer type, 다른 object의 포인터 또는 void 포인터를 제외한 어떤 type간에도 변환이 일어나선 안 됨
NUREG_053	Pointer type과 integral type간에 cast 금지
NUREG_054	Object type을 가리키는 pointer를 다른 object pointer type으로 변환 금지
NUREG_055	integer type 변수에 대하여 부호 변환 금지
NUREG_056	implicit type conversion 금지
NUREG_057	서로 다른 name space에서 같은 이름의 identifier를 정의해서는 안됨 (단, structure와 union member 이름은 제외)
NUREG_058	객체나 함수가 한번 이상 선언된다면, 그 type은 호환 가능해야 함
NUREG_059	float 타입 사용 금지
NUREG_060	Floating-point expression은 equality 또는 inequality 검사에 사용 금지
NUREG_061	나눗셈에 음수가 사용되는지 검사
NUREG_062	% 연산자를 쓸 때 나머지가 양수라고 가정하지 마라
NUREG_063	integer type 변수에 대하여 부호 변환 금지
NUREG_064	우선 순위가 다른 연산자 혼용 시 괄호를 사용하였는지 검사
NUREG_065	논리 연산자 && 와 의 오른쪽 피연산자에 side effect 포함 금지
NUREG_066	condition에서 가장 상위에 assign 연산자 사용 금지 : && 연산의 피연산자, 연산의 피연산자, top-level에 assign 연산자 사용금지
NUREG_067	다중 상속 금지
NUREG_068	가상 함수 정의 금지
NUREG_070	동적할당이 없는 포인터에 free 금지
NUREG_071	중복 free 금지
NUREG_072	errno를 사용하는 라이브러리 함수를 호출하기 전에 errno 값을 0으로 설정
NUREG_073	수학함수가 안전하게 사용되었는지 검사
NUREG_074	파일 입출력 사이에 플러시나 파일 위치 조정 함수가 존재하는지 검사

규칙	제목
NUREG_075	FILE 포인터에 대해 중복 close 금지
NUREG_076	signed 정수 연산이 오버플로되지 않도록 보장해야 한다.
NUREG_077	division by zero가 발생하면 안된다.
NUREG_079	Switch, while, do ... while, for 문의 body는 복합문이어야 함
NUREG_080	제어문의 brace 위치 검사
NUREG_081	indentation 규칙
NUREG_082	External identifier가 식별 가능한지 검사
NUREG_083	중첩된 C 스타일 주석 사용 금지
NUREG_084	C 스타일 주석만 사용해야 함
NUREG_085	주석 스타일이 혼용되었는지 검사
NUREG_086	함수의 코드 라인 수(LOC) 제한
NUREG_087	question 연산자 사용 금지
NUREG_088	조건문에서 bitwise 연산자 (&,) 사용 금지
NUREG_089	pointer to member(function/field) 사용 금지
NUREG_090	#include 앞에는 다른 preprocessor directive나 주석만 허용
NUREG_091	external 함수 선언이 헤더에 있는지 검사
NUREG_092	매크로 정의시 괄호 및 꺾쇠(brace, parenthesis, bracket) 짝이 맞는지 검사
NUREG_093	함수 매크로의 정의에서 각 parameter들은 괄호에 의해 감싸져야 함(단, #이나 ##의 피연산자로 사용하는 경우는 제외)
NUREG_094	숫자 상수 매크로 정의 금지 (const identifier 권장)
NUREG_095	extern 객체나 함수는 오직 하나의 파일에서만 선언되어야 함
NUREG_096	매크로 함수 파라미터 개수 제한
NUREG_097	클래스 연산자가 private으로 선언되었는지 검사
NUREG_098	default parameter 선언금지
NUREG_099	virtual 함수를 상속받은 함수에 virtual 속성을 명시하였는지 검사
NUREG_100	asm 문장 사용 금지
NUREG_101	C 코드에서 C++ keyword 사용 금지
NUREG_102	표준 library에 있는 매크로, 객체, 함수들의 이름의 재사용 금지

규칙	제목
NUREG_103	undersocre나 double underscore로 시작하는 타입명 사용 금지
NUREG_104	undersocre나 double underscore로 시작하는 함수명 사용 금지
NUREG_105	undersocre나 double underscore로 시작하는 변수명 사용 금지

표 15. 코딩 규칙 - NUREG/CR-6463



CWE_C(v2.8)

규칙	제목
CWE-119	메모리 버퍼의 경계에서 부적절한 제한이 발생하는 연산 수행 검사
CWE-120	입력 값의 크기를 검사하지 않고 버퍼를 복사하였는지 검사
CWE-121	스택 기반에서 버퍼 오버플로가 발생하는지 검사
CWE-122	힙 기반에서 버퍼 오버플로가 발생하는지 검사
CWE-129	검증된 배열 인덱싱인지 검사
CWE-131	버퍼 사이즈에 대해 잘못된 계산이 있는지 검사
CWE-135	멀티 바이트로 구성된 문자열에 대해 잘못된 길이 계산이 있는지 검사
CWE-14	버퍼값을 삭제하는 코드가 컴파일러에 의해 제거되는지 검사
CWE-170	널 종료 검사
CWE-190	정수 오버플로 또는 wraparound 검사
CWE-192	강제 정수 변환 검사
CWE-193	off-by-one 오류 검사
CWE-20	입력값이 검증되었는지 검사
CWE-242	위험이 내재하는 함수 사용 금지
CWE-252	함수의 반환값을 확인하였는지 검사
CWE-311	중요한 정보를 암호화 하였는지 검사
CWE-327	안전하지 않은 암호화 알고리즘을 사용하였는지 검사
CWE-330	적당하지 않은 무작위 수 사용 검사
CWE-369	Divide by zero 검사
CWE-390	함수의 반환값에 대해 오류 검증을 하였는지 검사
CWE-391	오류 검증 여부 검사
CWE-416	메모리가 해제된 객체가 사용되었는지 검사
CWE-456	초기화 누락 검사
CWE-464	자료 구조 sentinel 추가 검사
CWE-466	pointer type과 integer type 간 명시적 변환 금지
CWE-467	sizeof()의 인자로 포인터 변수 사용 금지
CWE-469	사이즈 결정을 위한 포인터 뽐셈 연산 사용
CWE-476	메모리 할당 함수의 반환값이 검증되지 않아 null 포인터 참조가 시도되는지

규칙	제목
	검사
CWE-480	잘못된 연산자 사용 검사
CWE-561	도달 불가능한 코드 사용 금지
CWE-562	스택 변수의 주소 반환 금지
CWE-628	함수 호출의 인자가 정확한지 검사
CWE-665	부적절한 초기화 검사
CWE-676	잠재적으로 위험성을 가진 함수 사용 검사
CWE-681	Numeric 타입 사이의 부정확한 변환 검사
CWE-682	부정확한 계산 검사
CWE-686	함수 호출의 인자로 부정확한 타입 사용 금지
CWE-687	함수 호출의 인자로 부정확한 값 사용 금지
CWE-704	잘못된 변환 금지
CWE-705	잘못된 제어 흐름 금지
CWE-754	비정상적이거나 예외적인 조건에 대해 잘못된 검증이 발생하는지 검사
CWE-768	잘못된 short-circuit 평가 검사
CWE-788	버퍼를 넘어서는 접근 검사
CWE-805	잘못된 길이 값으로 버퍼에 접근하는지 검사

표 16. 코딩 규칙 - CWE_C

IEC 62279

규칙	제목
IEC 62279_B.11_01	변수 사용 전 값 할당 검사
IEC 62279_B.11_02	미사용 변수 검사
IEC 62279_B.11_03	기록한 변수를 읽었는지 검사
IEC 62279_B.15_01	특정 함수 내에서 특정 파라미터를 사용하기 전 검사
IEC 62279_B.15_02	함수의 모든 파라미터가 읽기 전용인지 검사
IEC 62279_B.15_03	문자 상수는 반드시 쓰기 접근이 불가능해야 함
IEC 62279_B.15_04	함수의 반환값이 검증되었는지 검사 (함수 호출 인자로 사용된 경우 검증으로 간주 안함)
IEC 62279_B.16_01	goto 문장 사용 금지
IEC 62279_B.16_02	함수의 직간접적인 재귀호출 금지
IEC 62279_B.16_03	comment 비율 검사
IEC 62279_B.2_01	파일의 코드 라인 수(LOC) 제한
IEC 62279_B.2_02	모듈의 수행 가능한 경로의 수 제한
IEC 62279_B.2_04	함수의 최대 nesting depth 제한
IEC 62279_B.42_01	함수 복잡도(cyclomatic complexity number) 제한
IEC 62279_B.43_01	함수의 코드 라인 수(LOC) 제한
IEC 62279_B.43_02	함수가 하나의 exit point 를 가졌는지 검사
IEC 62279_B.43_03	함수의 파라미터 개수 제한
IEC 62279_B.4_01	division by zero를 방지하기 위한 제수 검증 여부 검사
IEC 62279_B.4_02	변수의 타입 크기를 벗어나는 상수 할당금지
IEC 62279_B.62_01	동적 메모리 할당 금지
IEC 62279_B.62_02	반복문에는 loop를 종료하기 위해 최대 하나의 break 문이 사용되어야 함
IEC 62279_B.62_03	변수 선언 시 초기화 여부 검사
IEC 62279_B.9_01	함수 내 도달 불가능한 코드 사용 금지
IEC 62279_B.9_02	swith문 안에 case, default 절이 아닌 문장이 존재하면 안됨
IEC 62279_B.9_03	Switch label(case, default)을 포함하는 가장 가까운 문장이 switch 문인지 검사
IEC 62279_B.9_04	문장이 있는 모든 switch 절이 break문으로 끝나는지 검사

규칙	제목
IEC 62279_B.9_05	switch문에 default절 존재 검사
IEC 62279_B.9_06	switch 의 default 절은 마지막 절에 위치하는 지 검사
IEC 62279_B.9_07	switch문이 하나 이상의 case 문을 가졌는지 검사
IEC 62279_B.9_08	비교 조건식의 연산 결과가 항상 같게 나오는 표현식 사용 금지

표 17. 코딩 규칙 - IEC 62279